МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

В. Л. БЫКОВ, Ю. П. АШАЕВ

Основы информатики

Допущено Советом Брестского государственного технического университета в качестве пособия для студентов технических специальностей

БРЕСТ 2006

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

Рецензент:
профессор кафедры "Вычислительные методы и программирование" Белорусского государственного университета информатики и радиоэлектроники **Колосов С. В.**

В. Л. Быков, Ю. П. Ашаев.
Б95 Основы информатики. Пособие. Издательство БГТУ. – Брест, 2006. – 430 с.: ил.

ISBN 985-493-032-7

Пособие разработано в соответствии с типовой программой по дисциплине "Информатика", утвержденной Министерством образования Республики Беларусь. Излагаются общие сведения об информатике, как основе познания, и ее категориях, системах счисления, применяемых в вычислительной технике, об истории развития вычислительной техники, устройстве компьютера и его составных частей, программном обеспечении: операционная система Windows 98, сервисная оболочка Windows Commander, программы архивации, приведены сведения об антивирусных программах. Изложены общие сведения о работе в вычислительной системе MathCad. Достаточно подробно рассмотрены вопросы алгоритмизации и программирования в среде Visual Basic 6.0. Приведены основные сведения о приложениях Windows: Word, Excel. Даны краткие сведения о сетевых технология. Предназначено для студентов очной и заочной форм обучения технических специальностей, может быть полезно аспирантам, магистрантам, а также всем, кто самостоятельно изучает основы информатики и вычислительной техники.

УДК 004(75)
ББК 22.183.492 : 73Я73

ISBN 985-493-032-7

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.

Учитывая роль информатики в современном обществе, 6 сентября 1995 года в Республике Беларусь был принят специальный закон "Об информатизации". В этом законе даны основные термины и определения, определена государственная политика в сфере информатизации, определены принципы информатизации и правовые отношения в сфере информатизации. А в 1998 году постановлением Совета Министров № 129 от 29 января одобрена Республиканская программа "Информатизации системы образования".

Целью дисциплины "Информатика" 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 является базовое программное обеспечение, операционная система, сервисные оболочки, язык программирования Visual Basic, сетевое программное обеспечение, прикладное программное обеспечение: пакеты прикладных программ Word и Excel, математическая система MathCad.

Чем вызвана необходимость написания данного учебного пособия, при том, что учебников и учебных пособий по данной дисциплине достаточно много. Это объясняется следующим: большинство учебных пособий написано применительно к экономическим и гуманитарным специальностям. Основное внимание в них уделяется прикладному программному обеспечению – Word, Excel, сетевое программное обеспечение, подготовке презентаций и в то же время вопросы алгоритмизации, программирования не находят в этих изданиях должного отражения. Поэтому авторы надеются, что изданием данного учебного пособия удастся восполнить указанный пробел.

Предмет "Информатика" настолько обширен, что в рамках одного учебного пособия невозможно подробно изложить все его аспекты. Поэтому авторы придерживались типовой программы, утвержденной Министерством образования Республики Беларусь. Однако ограниченность времени, отводимого на изучение предмета, поставила перед авторами альтернативу: либо изложить понемногу сведения о всех приложениях Windows, либо сосредоточить внимание на наиболее важных, с нашей точки зрения, для студентов технических специальностей приложениях. Мы пошли по второму пути.

Пособие состоит из двух книг. В первой книге излагаются теоретические вопросы и приводятся необходимые примеры для иллюстрации изучаемого материала. Вторая

часть - практикум. В ней приведены задания для практические занятий по приобретению и закреплению практических навыков использования компьютерной техники.

Первая книга состоит из семи разделов.

В первом разделе приведены основные понятия об информации и информатике, даны основные термины и определения, форматы представления информации, дано понятие о системах счисления, истории и перспективах развития вычислительной техники, устройстве, принципе работы персонального компьютера и его составных частей, приведена классификация программного обеспечения, приведены краткие сведения об операционной системе Windows.

Во втором разделе приведено описание математической системы MathCad и рассмотрены основные приемы работы в ее среде.

Третий раздел посвящен алгоритмизации и программированию. Базовые структуры языка программирования и алгоритмы решения типовых задач рассматриваются на базе языка программирования Visual Basic 6.0. Язык программирования Visual Basic 6.0 выбран по ряду причин. Этот язык прост в изучении. Синтаксис большинства его операторов совпадает со структурами алгоритмического языка программирования, изучаемого в школах. Кроме того, этот язык достаточно развит и приближается по своим возможностям к таким языкам программирования высокого уровня, как Pascal, Delphy, C. Язык Visual Basic является объектно-ориентированным языком программирования и позволяет создавать 32 – х разрядные приложения для работы в среде Windows. Следующим и весьма важным аргументом в пользу выбора языка программирования VB 6.0 в качестве базового языка программирования для студентов технических специальностей не программистов является то, что различные версии этого языка используются во всех стандартных приложениях Windows, таких как, например, Word, Excel, Access для разработки пользовательских процедур и функций для автоматизации управления разрабатываемыми приложениями.

В четвертом разделе дано описание среды разработки приложений языка программирования Visual Basic и рассмотрены подробно вопросы программирования в его среде.

Пятый раздел посвящен текстовому процессору Microsoft Word 2000. Основное внимание уделено оформлению документов средствами текстового процессора.

В шестом разделе приведены основные сведения об электронной таблице Microsoft Excel и использовании ее для решения инженерных задач.

Седьмой раздел посвящен сетевому программному обеспечению. Излагаются основные понятия о глобальной сети Internet, протоколах и адресации. Рассмотрена структура браузера Internet Explorer и назначение его основных элементов. Рассмотрены принципы работы электронной почты.

Пособие подготовлено к изданию Быковым Вячеславом Леонидовичем. Разделы 2 и 7 подготовлены Ашаевым Юрием Павловичем.

Пособие предназначено для студентов технических специальностей, может быть полезно и для студентов других специальностей, а также для всех желающих самостоятельно изучить основы дисциплины "Информатика".

Авторы выражают благодарность и признательность сотрудникам Брестского государственного технического университета, оказавших помощь в подготовке отдельных разделов настоящего пособия: к.ф-м.н. доценту Ракецкому В. М., к.т. н. доценту Костюку Д. А., начальнику отдела информационных технологий Цыганку В. В.

Особую благодарность авторы выражают рецензенту - профессору кафедры "Вычислительные методы и программирование" Белорусского государственного университета информатики и радиоэлектроники Колосову С. В. за ценные замечания, способствовавшие улучшению структуры и содержания пособия.

1. ОБЩИЕ СВЕДЕНИЯ О ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ

1.1. ОСНОВНЫЕ ПОНЯТИЯ И КАТЕГОРИИ ИНФОРМАТИКИ

1.1.1. 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. В законодательстве дается следующее определение информации[1]: «***информация*** - ***сведения о лицах, предметах, фактах, событиях, явлениях и процессах»***.

Информация, как объект научного исследования, стала рассматриваться лишь во второй половине 20 столетия, хотя информацией человечество пользовалось во все времена. Информация имеет ряд важных характеристик: структуру, форму, 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 – 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 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 моря и слышит шум прибоя. Морской прибой – 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. 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.

[1] Определения даются в соответствии с Законом Республики Беларусь от 6 сентября 1995 г. № 3850 -XII Об информатизации.

Собранная информация обрабатывается, анализируется, обобщается и накапливается на тех же носителях 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 ***документированную информацию, циркулирующую в процессе ее обработки на электронно-вычислительных машинах.***

Данные хранятся в базах данных. ***База данных – совокупность взаимосвязанных данных, организованных по определенным правилам на машинных носителях.*** Базы данных могут объединяться в банки данных. ***Банк данных – организационно-техническая система, включающая одну или несколько баз данных и систему управления ими.***

Знания – это информация, на основании которой путем логических рассуждений могут быть получены определенные выводы. Для обобщения накопленных знании и обеспечения возможности использования их в практической деятельности, они также как и данные организуются в базы знаний. ***База знаний - совокупность формализованных знаний об определенной предметной области, представленной в виде фактов и правил.***

Рис.1.1 Цикл получения информации

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, 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* – организационный социально-экономический и научно-технический процесс обеспечения потребностей органов государственной власти, юридических и физических лиц в получении сведений о лицах, предметах, фактах, событиях, явлениях и процессах на базе информационных систем и сетей, осуществляющих формирование и обработку информационных ресурсов и выдачу пользователю документированной информации.**

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*** – ***совокупность методов, способов, приемов и средств обработки документированной информации, включая прикладные программные средства, и регламентированного порядка их применения.*** 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, включающая 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 двадцать превом веке 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 информация?
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?

1.1.2. ИНФОРМАТИКА

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 преда-

ний. 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, но и "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 информатика. 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:

I) разработка автоматизированных 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.

СИСТЕМЫ СЧИСЛЕНИЯ

Для хранения и передачи информации человечество на протяжении своего исторического развития разработало различные системы знаков. Но машина не понимает эту систему знаков. Ее этому надо "научить". В силу конструктивных особенностей цифро-

вые ЭВМ способны различать сигналы разного уровня в виде нулей и единиц, которые можно использовать для кодирования информации. Для представления данных в ЭВМ получили применения различные 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, L, C, D, M. 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 A можно записать:

$$A(q)=a_{n-1}q^{n-1}+a_{n-2}q^{n-2}+\ldots+a_{i}q^{i}+\ldots+a_{1}q^{1}+a_{0}q^{0}+a_{-1}q^{-1}+\ldots+a_{-m}q^{-m}, \qquad (1.1)$$

где a_i - символы алфавита системы счисления;

q - основание системы счисления;

n, m - число разрядов в целой и дробной части соответственно.

Пример 1.1 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 преимущественно позиционные системы счисления, приведённые в табл. 1.1.

Таблица 1.1 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

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

$$n=log_q(A_{max}(q)+1). \quad (1.2)$$

И наоборот, если известна длина разрядной сетки n, то можно определить максимальное число $A_{max}(q)$, которое можно представить с использованием данной разрядной сетки:

$$A_{max}(q) = q^n-1. \quad (1.3)$$

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. и 1.3 .

Таблица 1.2 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 Код: 11101	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 числа .01011

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:

Таблица 1.3 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.	

$$75I254_8 = 111\ 101\ 001\ 010\ 101\ I00_2$$

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

$$1100\ 0111\ 1011\ 0101_2 = C7B5_{16},$$

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

$$AE3C_{16} = 1010\ 1110\ 0011\ 1100_2,$$

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.).

Источник информации служит для измерения параметров контролируемой среды и

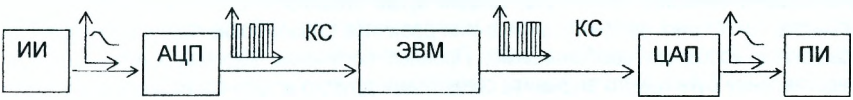


Рис. 1.2. Передача информации

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: AAB, 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 (американский стандарт кодов для обмена информацией). В настоящее время в приложениях операционной системы Windows применяется UNICOD, который позволяет закодировать 65536 символов.

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, 8 байт для 64 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.

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. 1.3.).

Рис.1.3. 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 А, P_A - характеристика числа А, 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 находиться в пределах $q^{-1} \leq |m_A| < 1$.

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. Распределение разрядов при 32-х разрядной сетке приведено на рис. 1.4.

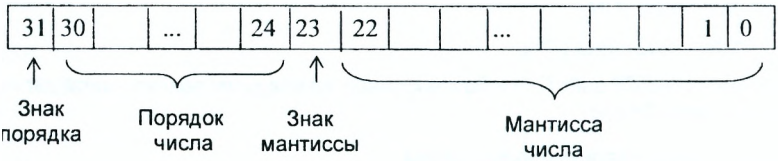


Рис.1.4. Формат представления числа с плавающей точкой

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 азбуки, 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. Назовите основные этапы развития информатики.
2. Дайте определение понятию информатика.
3. Какие основные задачи призвана решать информатика как наука.
4. Что такое основание системы счисления?
5. Запишите число 476,15 в общей форме записи чисел в позиционной системе счисления.
6. Составьте таблицу двоичных кодов шестнадцатеричной системы счисления, используя общую форму записи чисел в позиционной системе счисления.
7. Переведите в восьмеричную систему счисления код 01110011111001_2.
8. Переведите в шестнадцатеричную систему счисления код $1011\ 0110\ 1111\ 1010_2$.
9. Переведите код $1001\ 1000\ 0101\ 0010_{2\text{-}10}$ из двоично-десятичной системы счисления в десятичную систему счисления.
10. Переведите число $AC37_{16}$ в двоичную систему счисления.
11. Переведите в двоичную систему счисления число 375_{10}. Сделайте проверку.
12. Переведите в двоичную систему счисления число $0{,}761_{10}$ с точностью до 0,01. Сделайте проверку.
13. Переведите из двоичной системы счисления в десятичную систему счисления следующие коды: 10110110_2, 0.10110011_2.
14. Изобразите общую структурную схему передачи сообщений.

15. Что представляет собой информационное сообщение, в каком виде оно передаётся?
16. Опишите принцип работы системы передачи информации.
17. Что такое код? Какое количество кодовых комбинаций необходимо для кодирования всех символов русского алфавита, белорусского алфавита?
18. Что такое разрядная сетка ЭВМ? Чем она определяется?
19. Какие форматы используются для представления: а) целых; б) вещественных чисел?
20. Сколько кодовых комбинаций можно представить с помощью одного байта?
21. В чём состоит особенность представления графической информации?
22. Что такое фонема? Сколько фонем необходимо для представления звуков русского алфавита?
23. Какие устройства используются для воспроизведения речи?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с основными понятиями предмета информатика, историей, способами и форматами представления информации в ЭВМ. Основным способом представления информации в цифровых вычислительных машинах является алфавитный способ, а для кодирования информации применяется двоичный код. Форматы представления информации зависят от характера этой информации. Для представления символов алфавита и спецзнаков используется один байт, для представления чисел – два, четыре или восемь байтов.

В качестве основной системы счисления в ЭВМ используется двоичная система счисления. Но наряду с ней используются также восьмеричная, шестнадцатеричная.

1.2. ИСТОРИЯ И ПЕРСПЕКТИВЫ РАЗВИТИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Электронно-вычислительная машина (ЭВМ) представляет собой устройство, предназначенное для автоматической обработки информации в процессе решения вычислительных и информационных задач.

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).

1.2.1. 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.

1.2.2. МЕХАНИЧЕСКИЙ ЭТАП

Около 1594г. *Джон Неппер* 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.

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.

1.2.3. ЭЛЕКТРОМЕХАНИЧЕСКИЙ ЭТАП

Этому этапу предшествовали такие выдающиеся открытия в области физики, как открытие электрона, изобретение электромагнита, электро-двигателя. 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.

1.2.4. 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.

Принято выделять пять этапов в развитии электронной вычислительной техники, связанных с развитием элементной базы[2] (табл. 1.4.):

- 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 *АВС*.

ПЕРВОЕ ПОКОЛЕНИЕ ЭВМ (С 1945 ГОДА)

Появлению первых ЭВМ предшествовали такие фундаментальные изобретения как изобретение электронной лампы (1879 г.). Изобретение триода (1913 г.). Триод, 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. Первая ЭВМ полностью на электронных лампах была названа ENIAC (ЭНИАК - электронный числовой интегратор и вычислитель). Она была изобретена Эккертом и Маучли и создана в США в 1946 году. Эта ЭВМ содержала 18000 электронных ламп и была в 1000 раз более быстродействующей, чем АВС: 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 электрон-

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

ная 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 (накапливания, 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 совершенствованием технических характеристик ЭВМ 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).

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.

Таблица 1.4 . 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 – триод.	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	1948 – 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.
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	Новые языки программирования: Бейсик – 1964, ПЛ/1 – 1964. 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.			

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 секунду.

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 большое 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 супер ЭВМ 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 количества 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 в целом. 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 сильно подорвало национальную электронную промышленность. 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.

В настоящее время, учитывая большое экономическое и военное значение развития собственной электронной промышленности, в союзном государстве Россия-Белоруссия разработана программа "СКИФ", направленная на создание собственной компьютерной техники. В результате выполнения этой программы разработан суперкомпьютер "СКИФ-К-500", а затем "СКИФ-1000", который по своим показателям входит в первую сотню современных суперкомпьютеров и продолжает "набирать обороты". Последние результаты, полученные по программе "СКИФ", дают России и Белоруссии большие преимущества и в политическом и в экономическом плане.

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 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 космосе, 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.

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с историей и перспективами развития вычислительной техники. Изменения поколений вычислительной техники, достигнутые успехи и перспективы развития неразрывно связаны с научными исследованиями и открытиями в области физики, химии, прогрессом в технологии изготовления компонентов электронных схем, приборов и технических средств для изготовления этих компонентов.

1.3. УСТРОЙСТВО И РАБОТА ЭВМ

1.3.1. 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, миникомпьютеры, 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, 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, варьи-

ровать 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*.

1.3.2. СТРУКТУРА И ПРИНЦИП РАБОТЫ ЭВМ

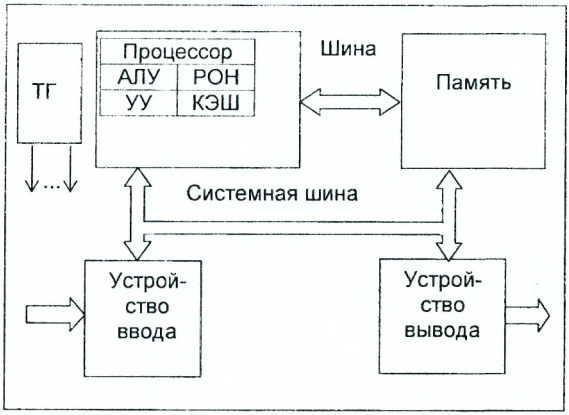


Рис.1.5. Упрощенная структура компьютера

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 фоннеймановской ЭВМ приведена на рис. 1.5.

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), регистры различного назначения (РОН) и 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.

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

ПАМЯТЬ

Память компьютера служит для хранения программ и данных. Она состоит из ячеек, связанных с шинами данных и адреса (рис. 1.6.).

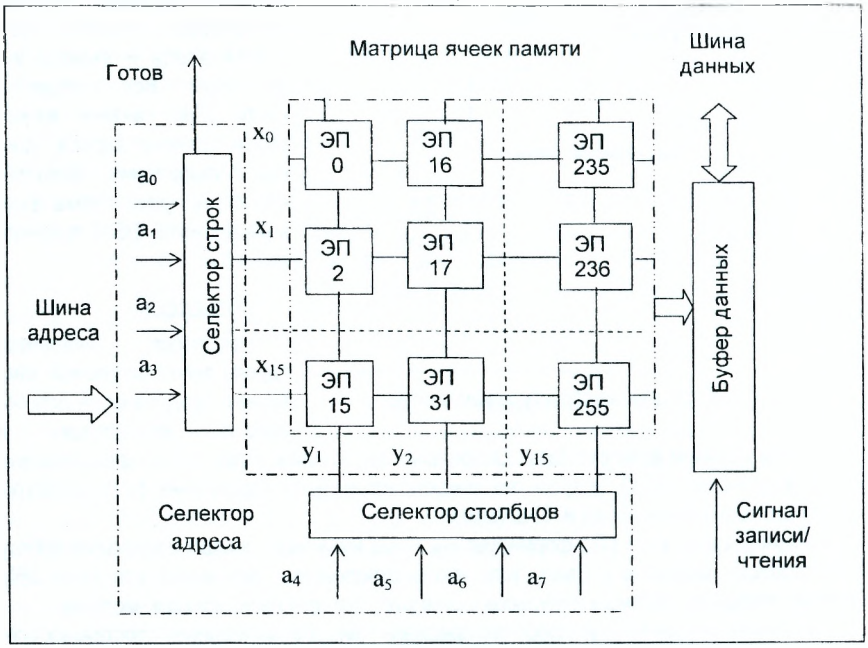


Рис. 1.6. Структура блока памяти

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 могут обеспечивать 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.

Устройства ввода-вывода состоят из механической части – собственно устройство, и электронного компонента, который называют ***контроллером*** или ***адаптером***. Контроллеры имеют обычно набор регистров для обмена информацией с центральным процессором или памятью. Адреса регистров устройств ввода- вывода принято называть ***портами***. В некоторых компьютерах адреса регистров устройств ввода-вывода являются частью физического адресного пространства процессора. В других компьютерах они образуют собственное адресное пространство за счет введения специальных операций ввода-вывода.

ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ

В ЭВМ используется *принцип программного управления*. Один из способов его реализации был предложен в 1945 г. американским математиком Д. Нейманом, и с тех пор

нейманов ский 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.

В процессе работы микропроцессора возможны прерывания. Прерывания возникают в результате неисправностей аппаратуры, запросов подключенных внешних устройств на обслуживание, запросов выполняемых программ на обслуживание или при возникновении ошибок вычисления. Прерывания принято делить на логические, программные и аппаратные. *Аппаратные прерывания* инициируются аппаратурой, например, сигналом от принтера, нажатием клавиши на клавиатуре, сигналом таймера. *Логические прерывания* возникают при нестандартных ситуациях в работе микропроцессора, например, деление на ноль, переполнение регистра, несоответствия типов данных, отсутствие затребованного файла и др. *Программные прерывания* инициируются программой пользователя, когда работающая программа хочет получить доступ к каким-либо ресурсам компьютера или другой программе. Каждое прерывание имеет уникальный номер и с ним связана определенная программа обработки прерывания. Процессор обрабатывает прерывания в соответствии с их приоритетом. Сигнал прерывания приостанавливает работу процессора, состояние обрабатываемого процесса запоминается и, в зависимости от кода прерывания, подключает модуль операционной системы, 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.

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, Celeron, Sempron фирмы Intel, Athlon, Duron фирмы AMD.

3. Разрядность. Разрядность ЭВМ определяется разрядностью процессора и характеризует точность вычислений и производительность машины. Различают 8-, 16-, 32- , 64- 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 подошла к порогу 4 ГГц.

5. Объем памяти компьютера определяет возможности ЭВМ по использованию современных пакетов прикладных программ. Максимальная оперативная память достигает 4096 Мбайт, КЭШ – память первого и второго уровней составляет 128 - 2048 Кбайт.

6. Ёмкость внешних запоминающих устройств (ВЗУ) определяет объём хранимой и используемой информации. Емкость накопителей на жестком диске достигает 400 Гбайт.

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

8. Массогабаритные характеристики.

9. Стоимость.

1.3.3. УСТРОЙСТВО ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

СОСТАВ БЛОКОВ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Персональные компьютеры выпускаются разными фирмами, но независимо от фирмы-производителя в состав современного ПК входят системный блок, видеомонитор, клавиатура, манипулятор типа "Мышь" (рис. 1.7). 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

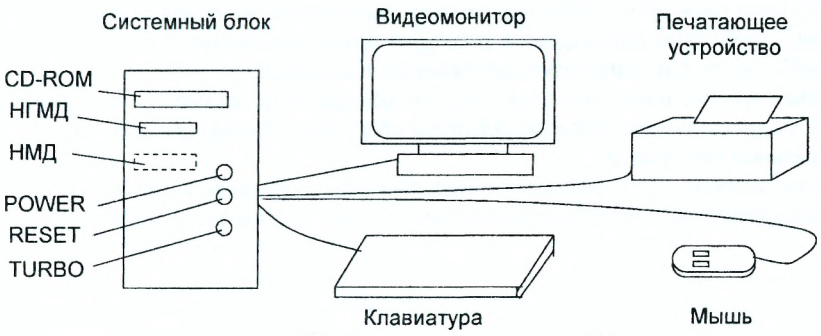


Рис. 1.7. Минимальная конфигурация ПК

блока (рис. 1.7) могут быть 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 обычно размещаются системная плата, платы адаптеров периферийных устройств, видеомонитора, накопитель на жестком диске (НМД), накопители на сменном диске (накопитель на гибком магнитном диске (НГМД), накопители на оптических дисках (CD-ROM)), блок питания.

Системная плата (материнская плата) представляет собой одноплатную микроЭВМ и может, при необходимости, функционировать самостоятельно. На ней размещены: микропроцессор, КЭШ–память второго уровня, основная память, 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, называемые ***слотами***, соединяется с выполненными на отдельных печатных платах модулями дополнительного оборудования – *адаптерами*.

На заднюю панель системного блока выведены разъёмы, через которые подключаются дополнительные устройства: дисплей, принтер, клавиатура, манипулятор мышь. Персональные компьютеры могут работать с периферийными устройствами, имеющими связь по параллельному интерфейсу *(Centronics)* или последовательному интерфейсу *(RS-232C, USB*). ПК может иметь один или два разъема для последовательного интерфейса.

Системная шина является сложным устройством, представляющим собой набор проводников и устройств сопряжения с процессором, памятью и другими устройствами. К системной шине могут подключаться также дополнительные платы расширения, например, аудио-платы для поддержки работы со звуком, сетевая карта для подключения к сети Интернет. Эти платы размещаются внутри компьютера.

Адаптеры также являются сложными электронными устройствами, в состав которых могут входить 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.

МИКРОПРОЦЕССОР

Основные характеристики

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. 1.5. К ним относятся тип микропроцессора, разрядность, тактовая частота, число команд. Плотность размещения элементов на кристалле (степень интеграции) характеризуется числом транзисторов, а также технологией изготовления. Под *технологией процессора* понимается наименьший размер одного элемента, например транзистора, диода, конденсатора. Технология изготовления процессора влияет и на его быстродействие: чем меньше размеры элементов и расстояние между ними, тем быстрее передаются сигналы в процессоре и элементах памяти. Разрядность шин данных и адреса также влияют на скорость работы процессора. Увеличение разрядности шин данных и адреса ускоряет процесс выборки данных, а, следовательно, повышает и скорость вычислений.

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 и Sempron с тактовой частотой до 2800 МГц, Pentium 4 и Athlon с частотой до 3500 МГц. Процессоры 8086, 80286 принято обозначать i86, а

Таблица 1.5 Характеристики процессоров

Модель (тип) 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	3800		0,18	42000	2004
Процессоры фирмы AMD							
K5	64	32	75-166		0,6-0,35		-
K6	64	32	166-233		0,35-,25		1997
Athlon	64	32	3500				2004

процессоры старшего ряда: 80386, 80486, Pentium – i386.

Быстродействие компьютера принято определять числом элементарных операций, выполняемых в единицу времени (оп/с). Быстродействие зависит от многих факторов и в первую очередь от тактовой частоты задающего генератора. Если первые ПК имели тактовую частоту 4, 8, 16 МГц, то в настоящее время частота тактового генератора подошла к порогу 4 ГГц 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, изготов-

ленный по 0,18 микронной технологии. А в настоящее время уже выпускаются процессоры данного типа с тактовой частотой 3,8 ГГц.

Кроме повышения тактовой частоты, увеличение производительности процессоров достигается путем разработки новых архитектур и алгоритмов обработки информации.

Имеются различные *архитектуры* процессоров. Под "архитектурой" процессора понимают конструкцию процессора и систему команд (инструкций). По способу представления команд различают два вида процессоров: процессоры типа *CISC* с полным набором команд и процессоры *RISC* с сокращенным набором команд. Процессоры второго типа нацелены на быстрое выполнение небольшого набора простых команд. Эти архитектуры постепенно сближаются, отбирая лучшие свойства каждой из архитектур.

К основным способам повышения производительности компьютера относятся:

суперскалярная архитектура;

конвейерная обработка информации;

использование разделенных КЭШ-памяти для команд и данных;

использование блока предсказания адреса;

использование блока вычислений с плавающей точкой;

поддержка многопроцессорного режима работы;

использование средств обработки ошибок.

Термин "*суперскалярная архитектура*" означает, что процессор имеет более одного вычислительного блока. Эти вычислительные блоки называются ***конвейерами.*** Наличие в процессоре двух и более конвейеров позволяет ему выполнять одновременно несколько команд. Каждый конвейер разделяет процесс выполнения команды на несколько этапов:

- 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 микропроцессоров идет не только по пути повышения разрядности и тактовой частоты, но и путем совершенствования *технологии обработки инструкций*, например, в микропроцессоре 386 операция сложения выполнялась за 6 тактов, а в 486-м – за два такта. Микропроцессоры следующих поколений выполняют операцию сложения за один такт. Для работы с видео-, аудио- и графическими данными Фирмой Intel разработана технология MMX - *технология обработки множественных данных одной инструкцией* (SIMD). Существенную роль играет технология *умножения тактовой частоты,* при которой скорость работы внутренних блоков в несколько раз выше работы остальных частей компьютера.

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

сопроцессор 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.

Управление памятью

Микропроцессоры, начиная с 80386, имеют аппаратные средства для управления памятью и могут функционировать в двух основных режимах: *реальной адресации* и *защищенной виртуальной*[3] *адресации*. В режиме реальной адресации процессор работает как быстрый 16-и разрядный процессор 8086 с расширенным набором команд и может выполнять только одну задачу. В защищенном режиме процессор использует весь механизм 32-х разрядной организации памяти, в том числе механизм поддержки виртуальной памяти и механизм переключения задач. При этом процессор может выполнять одновременно несколько задач. Для каждой задачи процессор i386 эмулирует виртуальные процессоры i86. Каждый виртуальный процессор управляет назначенным ему процессом независимо от других процессов. В защищенном режиме (этот режим обозначают V86) процессор i386 поддерживает страничную организацию памяти и средства многозадачности. При этом используются те же средства межзадачной защиты и защиты операционной системы от пользовательских задач, которые предусмотрены в процессоре i386. Максимальный размер виртуального адресного пространства для каждого виртуального процессора i86 составляет 1 Мб, в то время как физическое адресное пространство для процессора i386 составляет 4 Гб (2^{32}). Возможность микропроцессора работать в защищенном режиме позволила реализовать в операционных системах многозадачный режим работы.

В последующих моделях микропроцессоров фирмы Intel используются аппаратные средства поддержки многопроцессорных систем. Микропроцессор Merced имеет 64-разрядную архитектуру, содержит ряд новаторских решений, повышающих эффективность вычислений, такие как предсказание и спекулятивное выполнение.

Адресация памяти

Интерфейсная часть микропроцессора предназначена для связи и согласования микропроцессора с системной шиной компьютера, а также приема и предварительного анализа команд выполняемой программы и формирования полных адресов операндов и команд. Важным в этом плане является понятие порта ввода-вывода. ***Порт*** ввода-вывода

[3] виртуальный – вымышленный, не существующий реально, физически.

– это пункты системного интерфейса компьютера, через которые микропроцессор обменивается информацией с другими устройствами. Число портов у микропроцессора определяется разрядностью шины адреса. При 16-ти разрядной шине адреса число портов может быть 65536 (2^{16}), при 32-х разрядной шине адреса – 4 294 967 296 (2^{32}). Каждый порт имеет адрес – номер порта, соответствующий адресу ячейки памяти. При этом эта ячейка памяти может быть или частью основной памяти компьютера, или частью устройства ввода-вывода, использующего данный порт. Многие устройства (диски, клавиатура, принтер и др.) имеют постоянно закрепленные за ними порты ввода-вывода.

Внешний интерфейс

Отметим еще одну особенность микропроцессоров – внешний интерфейс (слот подключения). В зависимости от того, какой слот подключения имеет материнская плата, она может работать с определенным типом процессора. Например, на процессорах до Pentium III использовался интерфейс (форм-фактор) Slot 1, на последующих моделях - Slot 2. Pentium Pro имеет интерфейс Socket 8, который позволяет поддерживать до четырех микропроцессоров в симметричной мультипроцессорной системе. Во втором поколении процессоров архитектуры IA-64 применяется физический интерфейс Slot M.

ПАМЯТЬ ЭВМ

Классификация памяти персонального компьютера

Различают внутреннюю и внешняя 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* второго уровня. Внутренняя память, кроме ПЗУ, энергозависимая, то есть при выключении питания информация теряется. 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 на *электронных* внешних запоминающих устройствах.

Внутренняя память

Основными характеристиками памяти являются время доступа или максимальная частота, с которой может работать эта память, а также ее объем. *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 1,7 наносекунд (нс), а максимальная частота – более 600 МГц. *Объем* одного модуля памяти может составлять от 1 до 2048 Мбайт. На практике в настоящее время используются модули памяти 32, 64, 128, 512 Мбайт, 1 Гб и даже 2 Гб, например, DDR – 2 Gb PC3200 фирмы Kingston. Обычно на материнской плате можно размещать несколько модулей памяти, общий объем которых должен находиться в пределах доступного объема ОЗУ.

Постоянная память (ROM) обычно имеет объём 8 - 256 Кбайт. Микросхемы постоянной памяти устанавливаются на системной плате. Данные записываются в ПЗУ при изготовлении компьютера и хранятся в нём без изменений. В ПЗУ хранятся вспомо-

гательные 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 исправности ОЗУ, клавиатуры, наличия 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 от 8 Мбайт до 4 Гбайта. В ОЗУ информация загружается с началом работы ПК. Сначала туда записываются файлы операционной системы, которые приводят компьютер в состояние готовности к взаимодействию с пользователем. 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 (объемом) 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 позволяют создавать блоки памяти большой емкости.

КЭШ – память

Для повышения производительности компьютера на нем устанавливается до трех уровней КЭШ памяти L1, L2, L3. L1 – память первого уровня, расположена непосредственно в кристалле микропроцессора, объем ее достигает 1 Мбайта, L2 - КЭШ-память второго уровня, объем до 2 Мбайта. КЭШ – память второго уровня статическая. Эта память устанавливается вблизи процессора на системной плате и связана с ним скоростной шиной, работающей на более высокой частоте, чем остальная память, либо внутри картриджа микропроцессора (начиная с Pentium II, Pentium Pro). L3 – КЭШ-память третьего уровня. Она образована выделением и использованием некоторой части оперативной памяти специальными системными программами. Этот вид памяти используется для буферизации (предварительного хранения) данных при работе с жестким диском, CD-ROM и другими устройствами. Особенностью КЭШ-памяти является то, что она не доступна программистам. Управляет ей непосредственно микропроцессор.

Распределение оперативной и постоянной памяти компьютера

Постоянное запоминающее устройство и оперативное запоминающее устройство имеют единое адресное пространство, поэтому эту память принято называть ***основной*** памятью. Определенные области памяти компьютера имеют специальное назначение и используются отдельными командами и программами операционной системы.

Выделяют три основные области памяти: область таблицы векторов прерывания, область программ ОС, используемых при загрузке компьютера, рабочая область.

Область таблицы векторов прерывания определяет место расположения под-

программ для обработки прерываний. Прерывания – это запросы активных процессов и устройств ввода-вывода к процессору на обслуживание. Первые 1024 байта этой области резервируются для таблиц векторов прерываний, определяя тем самым 255 различных прерываний. Остальная часть этой области используется для запоминания состояния прерванного процесса. Эта область имеет абсолютные адреса памяти от 00000 до 00400 в шестнадцатеричном исчислении.

Область программ ОС, используемых при загрузке компьютера. Эта область является местом хранения программ постоянного модуля базовой системы ввода-вывода (BIOS). Так как эта система ввода-вывода контролирует основную работу компьютера и его частей, ей необходимо некоторый объем памяти для собственных записей, для ведения учета состояния системы. Среди этой информации имеется и область данных, удерживающая в памяти сигналы, посылаемые с клавиатуры, до тех пор, пока программа не начнет их обработку. Здесь же хранятся сведения об объеме памяти, характеристиках основного оборудования, установленного на компьютере, индикатор режима дисплея и многое другое. Область объемом в 256 байт предназначена специально для данных BIOS. Абсолютные адреса памяти этой области составляют диапазон от 00400 до 00500 в шестнадцатеричном исчислении.

Рабочая область хранит рабочие программы и данные.

Помимо основной памяти процессор может обращаться к памяти за пределами основной памяти компьютера. Эту область памяти принято называть *виртуальной* памятью.

Виртуальная память

Виртуальная память - это совокупность программно-аппаратных средств, позволяющих пользователю писать программы, размер которых превышает размеры имеющейся оперативной памяти. Виртуальная память решает следующие задачи:

- размещение данных в запоминающих устройствах разных типов;
- перемещение данных между запоминающими устройствами разного типа;
- преобразование виртуальных адресов в реальные адреса.

Способностью работать с виртуальной памятью обладают все процессоры, начиная с микропроцессора *Intel 80386*. Такое свойство микропроцессора, как управлять памятью, заставляет реальную память машины иметь рабочий адрес, отличающийся от истинного, физического адреса. Принцип использования виртуальной памяти приведен на рис. 1.8.

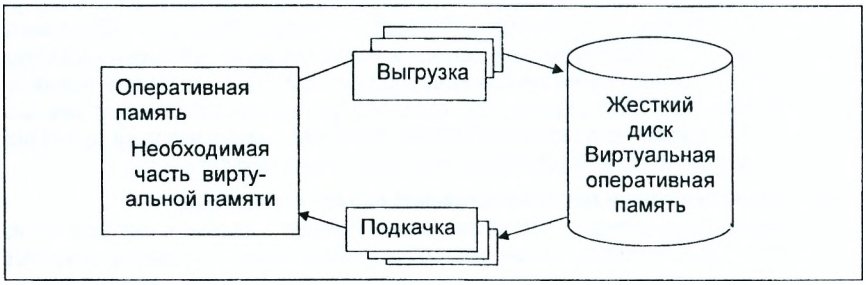


Рис. 1.8. Принцип использования виртуальной памяти

Если программа не помещается в оперативной памяти, ее разбивают на блоки, которые называют страницами, присваивают им адреса, которых нет в реальной памяти

компьютера, и размещают на разных носителях. Часть программы остается в оперативной памяти компьютера, а другая часть помещается на жесткий диск.

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 места 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. Этот процесс называется *подкачкой.*

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

Внешняя память

Накопители на магнитных дисках

В настоящее время наиболее широкое распространение в качестве внешних запоминающих устройств имеют ***накопители*** на магнитных дисках и магнитных лентах. Эти накопители предназначены для хранения, записи, чтения информации. ***Носителями*** информации в этих накопителях являются магнитные диски и магнитные ленты.

Принцип действия носителей на магнитных лентах и дисках 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.

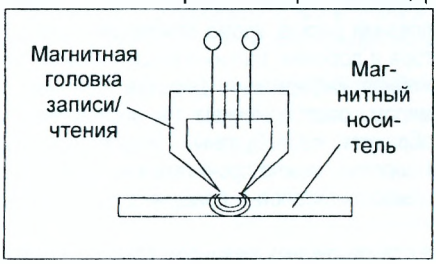


Рис.1.9. Принцип записи информации на магнитный носитель

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.1.9). 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.

На современных компьютерах устанавливаются, как правило, накопитель для гибких магнитных дисков (дискет) 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.

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

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 400 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.1.10). Вращение дисков и перемещение головок осуществляется с помощью двух электродвигателей.

Рис. 1.10. Принцип устройства жесткого диска

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 накопителей на магнитных дисках, определяющих их производительность, выделяют следующие: среднее время доступа, которое определя-

ется 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 166 Мбайт/с, скорость вращения диска от 3600 до 12000 об./мин.

Накопители на сменных гибких магнитных дисках. В настоящее время за счет использования современных технологий, таких как эффект Бернулли для бесконтактной записи, применение лазерного луча для позиционирования головок чтения-записи, технологии записи информации удалось значительно улучшить технические характеристики накопителей на гибких магнитных дисках. Так на дискетах размером 3,5" можно хранить до 1 Гбайт информации. Например, ***флоптические*** накопители имеют емкость больше 20,8 Мбайт. Внешне флоптические дискеты мало отличаются обычных 3,5-дюймовых дискет. На флоптические дискеты наносятся лучом лазера специальные серводорожки. Запись/чтение информации осуществляется с использованием процесса намагничивания. Одним из достоинств накопителей этого вида является то, что они могут читать и обычные 3,5-дюймовые дискеты емкостью 1,44 Мбайт и 720 Кбайт. Переносные накопители ZIP Drive фирмы IOMEGA хранят 100 Мбайт информации, а накопители JAZ этой же фирмы - 1 Гбайт информации. Запись и чтение информации в этих накопителях осуществляется лучом лазера. Накопители, использующие эффект Бернулли, поддерживают носители емкостью 35, 70, 90, 150 и 230 Мбайт.

Накопители на сменных жестких дисках. Накопители на сменных жестких дисках по внешнему виду мало чем отличаются от накопителей на гибких магнитных дисках. Размеры дисков -3,5 дюйма. Емкость носителя от 230 Мбайт до 2 Гбайт. Сменный магнитный носитель заключен в пластмассовый конверт, имеет металлическую основу и выполнен по технологии жестких дисков.

RAID-массивы. Для увеличения объема хранимой информации и повышения надежности промышленностью выпускаются RAID-массивы (Избыточный Массив на Дешевых Дисках). RAID-массив – устройство, состоящее из нескольких винчестеров и контроллера. Такое устройство обладает большим объемом дискового пространства, повышенной скоростью обмена информацией и значительной надежностью хранения информации. Надежность повышается за счет дублирования информации на несколько дисков.

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.

Накопители на оптических и магнитооптических дисках

Устройства для работы с оптическими дисками получили название CD-ROM. Они различаются по скорости считывания информации и возможности записи информации.

Накопители имеют следующую систему обозначений: 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. Основным показателем CD-ROM является скорость считывания информации. За единицу скорости считывания принята скорость чтения звуковых компакт-дисков – 150 Кбайт/с. У современных CD-ROM скорость чтения значительно выше, она указывается в обозначении диска. Например, Creative 24xMX означает 24-скоростная. Однако цифра 24 это не средняя, а максимальная скорость считывания. Средняя скорость считывания информации будет раза в полтора меньше.

В качестве носителей информации для оптических накопителей используются в настоящее время оптические (или лазерные) и магнитооптические диски. Они имеют размер 133 мм и 89 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.

Оптические диски. 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 дорожек 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 поразному 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. При записи информации в этом слое металла создается матрица хранимой информации лучом лазера.

В *перезаписываемых носителях* 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.

Магнитооптические диски с однократной записью и многократным считыванием выполняются из стекла, на поверхность которого наносится сплав из тербия, железа и кобальта, обладающий магнитными свойствами. Этот сплав имеет 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 прикладывается магнитное поле нужного направления. После остывания данный участок запоминает направление намагниченности. При нормальной тем-

пературе 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.

Известны накопители на магнитооптических дисках (APEX компании Pinnacle Micro) емкостью 4,6 Гбайт, использующие диски размером 5,25".

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.

Накопитель DVD

Накопители DVD предназначены для хранения и воспроизведения видео-, аудиозаписей высокого качества, компьютерной информации. Носителем информации является цифровой универсальный DVD-диск, но DVD- накопители могут читать и обычные CD-ROM-диски. Двухсторонние накопители DVD могут читать и CD-R- и CD-RW-диски. Цифровой универсальный диск по внешнему виду мало чем отличается от CD – дисков. DVD-диск имеет диаметр 120 мм. Он может быть односторонним или двухсторонним, на каждой стороне может быть один или два рабочих слоя. По этой причине эти диски имеют большую емкостью. Односторонние однослойные DVD имеют емкость 4,7 Гбайт, двухслойные – 8,5 Гбайт, двухсторонние однослойные – 9,4 Гбайт, а двухсторонние двухслойные носители могут вмещать до 17 Гбайт информации.

Электронные внешние запоминающие устройства

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.

Флэш-память (Flash – Drive) – новый вид внешнего запоминающего устройства, приобретающего все большую популярность. Другое название, встречающееся в литературе, и соответствующее маркировке ED – EasyDisk (буквально – легкий диск), например, ED 17, ED 80. Представляет собой полупроводниковое перепрограммируемое запоминающее устройство, предназначенное для хранения и переноса данных с одного компьютера на другой. Компактный, легкий, удобный и удивительно простой в эксплуатации. Как правило, имеет защиту от записи в виде переключателя на корпусе и программного пароля. Для его работы не нужны ни соединительные кабели, ни источники питания (включая батарейки), ни дополнительное программное обеспечение. Изготавливается в виде брелка с размерами в пределах до 90x25x13 мм. Объем памяти составляет от

16 Мбайт до 4 Гбайт. Подключается к контроллеру последовательной шины USB. Может работать под управлением операционных систем Windows 98 и старших версий Windows. В поставку входит диск с драйверами, удлинитель до 1 м, инструкция по эксплуатации. Широкое распространение сдерживается высокой ценой. По отношению Объем_памяти/Стоимость флэш-память значительно уступает жесткому диску. Со временем могут стать реальной альтернативой накопителям на гибких магнитных дисках.

ИНТЕРФЕЙС МАТЕРИНСКОЙ ПЛАТЫ

Для связи между собственными и подключаемыми устройствами материнской платы на ней выполняются шины и логические устройства, размещенные в микросхемах микропроцессорного комплекта – ***чипсета***. Наиболее распространенными как устаревшими так, и новыми устройствами являются шины стандартов ISA, EISA, VLB, PCI, FSB, AGP, PCMCIA. Основными их характеристиками являются максимальная скорость передачи данных, измеряемая в Мбайт/с, и число подключаемых устройств.

ИНТЕРФЕЙС ДЛЯ ПОДКЛЮЧЕНИЯ ВНЕШНИХ УСТРОЙСТВ

Для связи накопителей информации и других внешних устройств с компьютером также разработаны специальные интерфейсы. В настоящее время преобладают три вида интерфейса: IDE, SCSI и USB.

Интерфейс IDE (EIDE) используется для подключения накопителей на жестком диске. Поддерживает два способа обмена данными: через центральный процессор; обмен данными с ОЗУ без участия центрального процессора (прямой доступ к памяти). Максимальная скорость передачи данных 8,3 Мбайт/с.

Интерфейс SCSI имеет собственный контроллер и собственную BIOS, которая управляет шиной, освобождая центральный процессор. Стандарт Ultra 2 SCSI имеет 16 разрядную шину данных, позволяет подключать до 15 устройств и обеспечивает скорость передачи данных до 40 Мбайт/с.

В последнее время все большую популярность приобретает интерфейс *USB*. Он характерен тем, что позволяет подключать последовательно до 256 различных устройств, имеющих последовательный интерфейс. Поддержка стандарта Hi-Speed USB или USB 2.0 для современных системных плат стала обязательной. Hi-Speed USB - последний стандарт в этой области. Он обеспечивает скорость передачи данных до 480 Мбит/с (по другим источникам пропускная способность шины составляет до 1,5 Мбайт/с). Удобство шины состоит в том, что она практически исключает конфликты между различным оборудованием, позволяет подключать и отключать устройства без выключения компьютера и объединять компьютеры в локальную сеть без применения специального оборудования и программного обеспечения.

КЛАВИАТУРА

В настоящее время в эксплуатации находится значительное число типов клавиатуры, но все они построены по одному прин-ципу. На клавиатуре можно выделить условно четыре поля (рис.1.11):

1) *функциональное* - поле с функциональными и управляющими клавишами;

2) *алфавитно-цифровое* - поле, где расположены клавиши с буквами русского и латинского алфавита, цифрами, специальными

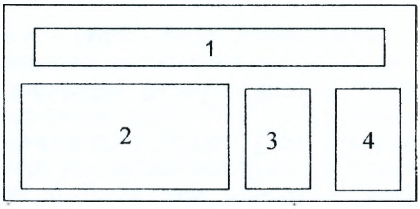


Рис. 1.11. Поля типовой клавиатуры

символами и некоторые управляющие клавиши:

3) *управления курсором* - поле с клавишами для управления курсором, вставки и удаления текста;

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

Назначение клавиш приведено в табл. 1.6.

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 CapsLock.

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

Наиболее часто при работе на компьютере используются следующие комбинации клавиш:

Ctrl+Alt+Del - перезагрузка операционной системы (горячий перезапуск);

Alt+PrScr - в операционной системе Windows эта комбинация клавиш помещает в буфер памяти копию активного окна Windows. При нажатии клавиши PrScr в буфер помещается копия всего рабочего окна.

Ctrl+Break - завершение работы выполняемой программы или команды;

Таблица 1.6 Назначение клавиш на клавиатуре IBM PC/AT

Клавиатура IBM PC/AT	*Назначение клавиш*
F1...F12	Функциональные клавиши
Caps Lock	Переключение на ввод 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
Prt Sc (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 (Insert)	Включение/выключение режима вставки символов
Del (Delete)	Удаление символа в позиции курсора
End	Перемещение курсора в конец файла
Home	Перемещение курсора в начало файла
PgUp	Перемещение курсора на страницу вверх
PgDn	Перемещение курсора на страницу вниз
← ↑ → ↓	Направление перемещения курсора

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

Ctrl+S - приостанавливает выполнение программы;

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

Shift+PrScr – печать графической копии экрана

Последние четыре комбинации клавиш используются только в среде дисковой операционной системы.

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 электронно-лучевой трубки (CRT-мониторы) и жидкокристаллические мониторы (LCD-мониторы).

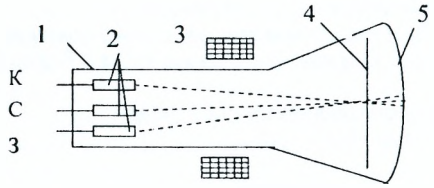


Рис. 1.12. Электронно-лучевая трубка

CRT- мониторы

Основу CRT-мониторов составляет электронно-лучевая трубка (ЭЛТ). Упрощенная структура ЭЛТ приведена на рис. 1.12. 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. 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. В цветных мониторах экран 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. 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. 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.

Режим работы монитора. Монитор может работать в двух режимах: текстовом и

графическом. При воспроизведении текста экран разбит условно на строки и столбцы. Число строк и столбцов зависит от разрешающей способности экрана. Нумерация строк и столбцов начинается с нуля. Для каждого символа отводится одно знакоместо. Имеется возможность программно изменять число строк и столбцов на экране. В графическом режиме экран представляется в виде набора точек – пикселей. Число точек по горизонтали и вертикали определяется разрешающей способностью экрана, которая может настраиваться пользователем.

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

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. 1.7).

Таблица 1.7 Рекомендуемые значения разрешающей способности монитора

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

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.

Глубина цвета или количество воспроизводимых одновременно цветов.

Современные мониторы способны воспроизводить практически неограниченное число цветов. Чаще всего устанавливаются палитры из 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 видеопа-

мяти. 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. Обычно она организуется в виде страниц. Видеомонитор имеет, как минимум, две страницы памяти: одна страница - видимая, отображается на экране, другая страница в это время заполняется. Смена кадра осуществляется во время обратного хода луча. Для качественного воспроизведения мультипликации компьютер должен иметь 4 страницы видеопамяти. Объем видеопамяти современных компьютеров составляет 4, 8, 16, 128 Мбайт и более.

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 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 три основные цвета. Комбинируя три основные цвета для каждой точки, можно воспроизвести любой цвет.

Несомненными достоинствами LCD-мониторов являются низкое потребление электрической энергии и отсутствие вредных излучений.

Разрешение LCD-мониторов одно, оно соответствует максимальному физическому разрешению CRT-мониторов. Однако имеется возможность выводить изображение на экран с меньшим разрешением либо путем использования части экрана (центрирование), либо растягиванием изображения на весь экран.

Несомненными достоинствами LCD-мониторов является 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 по-

дают высокое напряжение. Под воздействием этого напряжения в прилегающей к электроду газовой области возникает электрический разряд. Плазма газового разряда излучает свет в ультрафиолетовом 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. На плате видеоадаптера размещаются и микросхемы видеопамяти.

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 дешевым типом принтеров для персональных компьютеров. 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 секунд 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 принтерами 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. Скорость печати струйных принтеров составляет до 20 страниц за минуту.

Лазерные принтеры обеспечивают в настоящее время наилучшее (близкое к типографскому) качество печати. В них используется электрографический принцип создания изображения. Основными элементами лазерного принтера являются специальный барабан, источник лазерного излучения, электронная схема управления и сухой порошок – *тонер*. Тонер представляет собой кусочки железа, покрытые пластиком. Принцип рабо-

ты 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.

Скорость печати лазерных принтеров достигает 30 страниц за минуту.

К потребительским качествам принтеров относятся следующие показатели:

- качество и скорость печати;
- наличие русского шрифта в ПЗУ (аппаратная русификация) или возможность его загрузки из программы компьютера;
- надёжность;
- количество шрифтов, которые поддерживает принтер;
- возможность и удобство смены красящего элемента;
- совместимость с имеющимися программами по управляющим кодам.

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. Различают механические и оптические мыши. Механическая мышь 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. Оптическая мышь использует другие способы преобразования координат положения манипулятора. 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 коорди-

нат 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 монитора, 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) – оптико-электронное устройство, предназначенное для считывания графической и текстовой информации в компьютер. Текст вводится при этом в компьютер в виде графических символов. Распознавание текста осуществляется специальными программами, поставляемыми со сканером. При наличии соответствующего программного обеспечения, можно распознавать и рукописный текст. Сканеры бывают планшетные и ручные. Настольные сканеры обрабатывают весь лист бумаги целиком. Ручные сканеры необходимо проводить над нужным текстом или рисунком вручную.

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.

ЗАКЛЮЧЕНИЕ

Персональные компьютеры реализуют в основном фон-неймановский принцип программного управления, имеют открытую архитектуру, что позволяет пользователям самостоятельно менять конфигурацию компьютера путем подключения дополнительных устройств. Ядром вычислительной системы является процессор, который наряду с памятью определяет основные потребительские качества компьютера.

1.4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА

1.4.1. 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 *Windows Commander, Far Manager, Norton Commander, Volkov Commander* 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 поставляются обычно на дискетах драйверы всех основных устройств.

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). 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.

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.

Технологически-ориентированные ППП предназначены для управления технологическими процессами на фабриках, заводах, цехах.

1.4.2. ОПЕРАЦИОННАЯ СИСТЕМА

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, 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 9.x, 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*. Многонитевидная 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 соответствуют 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 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.

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. Каждая операционная система имеет свои правила доступа к устройствам. Ниже приведены правила для дисковой операционной системы и Windows.

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.

Файловая система

Файловая система – это часть операционной системы, которая обеспечивает пользователю удобный интерфейс при работе с данными, хранящимися на диске, совмест-

ное использование файлов несколькими пользователями и процессами. Основными задачами файловой системы является организация хранения, поиска, переименования, копирования, пересылки и удаления файлов и каталогов.

Файлы

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

Файлы бывают разных типов: обычные файлы, специальные файлы, файлы-каталоги. В файлах могут храниться тексты программ, документы, данные, сведения о файлах.

Обычные файлы в свою очередь подразделяются на текстовые и двоичные. Текстовые файлы состоят из строк символов, представленных в ASCII-коде. Это могут быть документы, исходные тексты программ, данные и т.п. Текстовые файлы можно прочитать на экране и распечатать на принтере. Двоичные файлы не используют ASCII-коды, они часто имеют сложную внутреннюю структуру, например, объектный код программы или архивный файл.

Специальные файлы - это файлы, предназначенные для управления устройствами ввода-вывода.

Файлы-каталоги – хранят сведения о файлах.

> *Файл — это организованный, поименованный набор данных на диске или другом машинном носителе.*

Имя файла

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

Имя файла в дисковой операционной системе может содержать от 1 до 8 символов. Имя должно начинаться с буквы и не должно содержать знаков пунктуации и пробелов, может содержать специальные символы: _ - $ # & @ ! % ( ) {} ' ~ ^. В операционной системе Windows для обозначения имен файлов можно использовать длинные имена. Это позволяет присваивать файлам вполне понятные имена, характеризующие их содержание. 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.

Расширение имени файла начинается с точки, за которой следует несколько символов. В дисковой операционной системе длина расширения имени файла фиксирована и не должна превышать трех символов. В операционной системе Windows расширение имени файла может содержать и большее число символов. Расширение имени файла не является обязательным, однако оно характеризует вид хранимой в нем информации. Основные расширения имен файлов приведены в табл. 1.8. Примеры имен файлов: *command.com, expert1.bas, autoexec.bat.* В имени фала *autoexec.bat*

autoexec - имя файла
.bat – расширение имени файла
autoexec.bat – полное имя файла.

Расширения имен файлов, приведенные в табл. 1.8, являются зарезервированными, то есть используются по умолчанию при работе с соответствующими программами.

Таблиц 1.8 Примеры использования типовых расширений имен файлов

Расширение	*Вид файла*
.COM, .EXE	программный файлы 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
.DOC	документ, подготовленный в редакторе Microsoft Word
.XLS	документ электронной таблицы Excel

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 так же, как и при вводе с клавиатуры.

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

В качестве маски используются символы * и ?. Символ * заменяет все имя файла, расширение имени файла или их часть, 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). Понятие "каталог" использовалось в дисковой операционной системе В операционной системе Windows вместо понятия каталог введено понятие *папка.* В дальнейшем не будем делать различия между этими понятиями.

Каталог - это с одной стороны - файл, в котором хранятся сведения о файлах и подчиненных каталогах. С другой стороны каталог – это группа файлов или каталогов, объединенных по какому-либо признаку: принадлежность одной программе, одному пользователю, имеющих один тип и т.п.

Каталог — это файл, в котором хранятся сведения о файлах и подчиненных каталогах.

К сведениям, которые хранятся в файлах, 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. Файл может иметь четыре атрибута: R - только для чтения, A – не архивирован, S - системный, H – скрытый.

На каждом диске всегда имеется один каталог, который называется корневым и обозначается символом "\" – обратный слэш. Все остальные каталоги размещаются в корневом каталоге. 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. 1.13) .

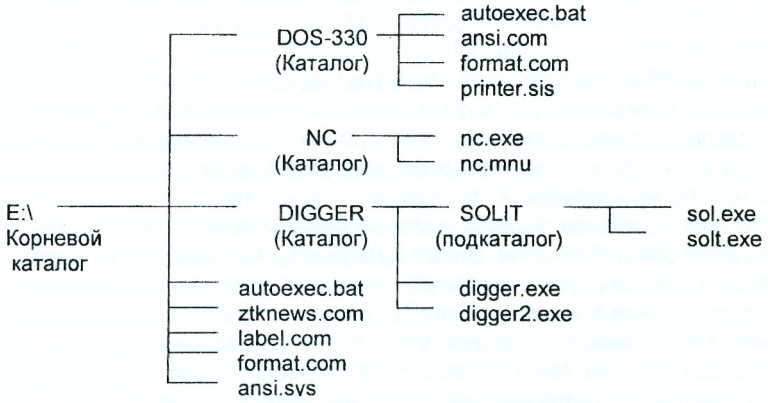


Рис.1.13. Дерево каталогов

Имена каталогов (папок) 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.

Путь (маршрут) – последовательность каталогов, разделенных 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.***

Число символов в пути не должно превышать 64 символа, а длина полного имени файла не должна превышать 255 символов.

Таблица размещения файлов

Сведения о файлах и каталогах хранятся в специальной области диска. В операционной системе Windows 9x (общее обозначение семейства ОС Windows 95, Windows 98) применяются файловые системы FAT-16, FAT – 32. В операционной системе Windows NT и Windows XP – файловая система NTFS.

В ОС Windows-95 используется FAT –16, в Windows-98 – FAT-16 и FAT-32. В этих файловых системах 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 обеих копиях была идентична.

Минимальным физическим объемом файла на дискетах является сектор одной дорожки. Его размер равен 512 байтам. Это минимальный размер адресуемого дискового пространства. На жестких дисках минимальный размер адресуемого дискового пространства может составлять несколько секторов. Эти группы секторов называют ***кластерами***. Размер кластера зависит от размера диска и от типа используемой файловой системы. 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 (устанавливать FAT-16 в разделы дисков больше 2 Гбайт не рекомендуется). Например, при объеме диска более 512 Мбайт размер кластера составит 64 Кбайта. Это значит, что если создать файл, содержащий всего один байт (логический размер), то на диске он займет 64 Кбайта (физический размер), то есть будет иметь место неэффективное использование дискового пространства. Другим ограничением является число секторов в кластере. Это объясняется следующим: для хранения числа секторов в кластере в FAT-16 отведен только один байт, следовательно, можно адресовать 255 секторов, но так как число секторов в кластере должно быть кратно 2, то максимальное число секторов в кластере ограничено 128 (64 Кбайта). Еще одним ограничением FAT-16 было то, что размер главного каталога был фиксирован и занимал на диске строго определенное место.

Файловая система FAT – 32 успешно работает с дисками больших размеров (до 2 терабайт). Но и в этой операционной системе при емкости диска 8 Гбайт размер кластера, согласно спецификации, составляет 4 Кбайта. В файловой системе FAT-32 сняты также ограничения на размер главного каталога. Одним из недостатков FAT-32 является то, что в оперативную память загружаются обе копии файловой системы и занимают

достаточно большой объем памяти, при 2-х гигабайтном диске – 4 Мбайта.

В связи с указанными недостатками файловых систем FAT-16 и FAT-32, в операционных системах Windows NT и Windows XP применяется файловая система NTFS. Она поддерживает диски объемом до 16 777 216 терабайт. Эта файловая система обладает большими преимуществами перед файловыми системами FAT-16 и FAT-32.К достоинствам файловой системы NTFS следует отнести следующее:

- каждый элемент NTFS – это файл. Самый главный файл, который имеет имя $MFT, представляет собой централизованный *каталог* всех остальных файлов диска. В системе NTFS нет понятия "главный каталог", как это было в FAT-16 и FAT-32. Отсутствуют ограничения на размер каталога. Файл $MFT – это и есть главный каталог;
- общая таблица файлов (MFT) имеет копию. Основная таблица размещается в начале диска, а копия – в середине. Это обеспечивает повышение надежности файловой системы при возникновении сбойных участков на диске в области размещения MFT-зоны. MTF-зона – это область диска, которую операционная система зарезервировала для размещения файла файловой системы.
- поддерживает кластеры разных размеров: от 512 байт до 64 Кбайт. Стандартом считается файл размером 4 Кбайта;
- обеспечивает более высокий уровень отказоустойчивости. Отказоустойчивость обеспечивается использованием ***транзакции*** (запрос на изменение данных). Механизм транзакции состоит в следующем: при выполнении операций записи, чтения действие должно быть совершено целиком или вообще не совершено. Если, например, при записи информации на диск обнаружено повреждение поверхности диска, то операция копирования откладывается полностью, неисправный участок помечается как сбойный, а данные записываются в другое место. То же произойдет при недостатке места на диске, например, при сохранении файла, уже имеющегося на диске. Операция будет отложена, данные на диске сохранятся в прежнем виде;
- обеспечивается высокая степень безопасности за счет разграничения прав доступа к файлам и папкам. Права доступа могут быть установлены для каждого пользователя, группам пользователей или и отдельным пользователям и большим группам;
- имеет встроенную поддержку сжатия дисков. Любой файл или каталог может храниться на диске в сжатом виде. Сжатие файлов имеет большую скорость. При этом часть файла может храниться в несжатом виде, а часть - в сжатом;
- позволяет зашифровать файлы или каталоги, обеспечивая безопасность важных данных, применяется 128 битовое шифрование файлов и папок.

Отличительной особенностью NTFS является то, что с увеличением размеров дисков эффективность хранения файлов на дисках при использовании NTFS не снижается. Файловая система NTFS обеспечивает большее быстродействие за счет кэширования файлов, а также большую эффективность использования дискового пространства. Кэширование файлов поддерживается для сетевых и совместно используемых дисков. Кэширование ускоряет доступ к файлам, а также позволяет пользователям работать с ними в автономном режиме.

Файловая система NTFS не совместима с файловыми системами FAT-16 и FAT-32. При установке на одном компьютере нескольких операционных систем допускается использовать две файловые системы, например FAT-32 и NTFS. Нельзя устанавливать на компью-

тер операционные системы, использующие разные версии файловой системы NTFS.

При работе с разделами жесткого диска, превышающими 32 Гбайта, рекомендуется использовать файловую систему NTFS.

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 процессор под управлением базовой системы ввода-вывода (BIOS), хранящаяся в ПЗУ, проверяет наличие памяти, установленных внешних устройств и проводит их инициализацию. Затем проверяются платы расширения, в том числе и видеоадаптера. После этого проверяется ОЗУ. Эта проверка может занять значительное время, до нескольких минут. 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. 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 в первом секторе нулевой дорожки блока начальной загрузки осуществляется загрузка 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, требуется перезагрузка компьютера. Процесс перезагрузки компьютера можно ускорить, если использовать "горячий" перезапуск, то есть выполнить загрузку ОС без выключения питания. При горячем перезапуске самопроверка компьютера не проводится и время загрузки ОС сокращается. Для "горячего" перезапуска компьютера необходимо нажать комбинацию клавиш ***Ctrl+Alt+Del.***

В тех случаях, когда перезапустить компьютер нажатием комбинации клавиш *Ctrl+Alt+Del* не удается, следует нажать клавишу RESET (перезапуск), установленную на лицевой панели системного блока.

Завершение работы. После окончания работы необходимо закрыть все приложения. Выход из программы Windows следует осуществлять только через команду главного меню ***Завершение работы***. В этом случае будет обеспечено сохранение всех настроек пользователя и сохранность данных. При выходе иным способом возможно нарушение целостности файловой системы и потеря данных. В таких случаях при повторном включении компьютера автоматически включается проверка дисков и восстановление файловой системы. Если пренебрегать проверкой дисков, то ошибки будут накапливаться и может наступить день, когда компьютер откажется нормально загрузиться и нормально работать. После ввода команды Завершение работы на экране монитора появляется окно диалога, в котором необходимо отметить мышью флажок "Выключить компьютер" и щелкнуть по кнопке ДА (ОК). Программа приступит к выполнению операций по подготовке к выключению питания.

Выключение компьютера

После завершения всех подготовительных операций к выключению питания на экран монитора выводится сообщение "Теперь питание компьютера можно отключить". В некоторых операционных системах выключение питания происходит автоматически. Мониторы часто включаются и выключаются независимо от системного блока. Поэтому не забудьте выключить питание видеомонитора.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что включает в себя базовое программное обеспечение?
2. Какие программные средства относятся к прикладному программному обеспечению?
3. Назовите классификационные признаки операционных систем.
4. Назовите наиболее известные операционные системы, применяемые на персональных компьютерах.
5. Что такое файловая система, для чего она предназначена?
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. Что такое атрибут файла, какие атрибуты имеет файл?
11. Что такое каталог? Какая информация в нем содержится?
12. Что такое спецификация файла? Приведите примеры.

ЗАКЛЮЧЕНИЕ

Программное обеспечение является второй составляющей, определяющей потребительские качества персонального компьютера. Оно определяет функциональные возможности компьютера по обработке данных. Ядром программного обеспечения является операционная система. В настоящее время доминирующее положение занимают графические операционные системы, которые обеспечивают пользователю дружественный интерфейс и берут на себя все функции по организации взаимодействия с внешними устройствами.

1.5. ОПЕРАЦИОННАЯ СИСТЕМА WINDOWS

1.5.1. ОБЩИЕ СВЕДЕНИЯ

Windows 98- графическая, многооконная, многозадачная с вытесняющей многозадачностью, 32-х разрядная однопроцессорная операционная система. Это одна из самых популярных ОС семейства Windows. В настоящее время она постепенно вытесняется операционной системой Windows XP. Но для обеспечения возможности использования большинства программ, разработанных под MS-DOS, чаще всего устанавливают две операционные системы: Windows 98 и Windows XP.

Операционная система Windows 98 поддерживает файловые системы FAT-16 и FAT-32.

Отметим главные особенности операционной системы.

Операционная система Windows 98 организована на ядре ОС MS-DOS, которое загружается в момент включения компьютера и активизирует графический интерфейс пользователя, а также обеспечивает полную совместимость с MS-DOS.

Вытесняющая многозадачность

Вытесняющая многозадачность обеспечивает возможность одновременного выполнения нескольких программ, переключения с одной задачи на другую, управления приоритетами выполнения программ. Операционная система самостоятельно, в зависимости от внутренней ситуации передает или забирает управление у того или иного приложения, не позволяя одному приложению занять все аппаратные ресурсы. Операционная система работает только в защищенном режиме. При этом каждой задаче (*процессу*) выделяется определенный участок памяти и никакая другая программа не может туда ничего записать, ни считать (эта особенность обеспечивается режимом защищенной виртуальной адресации процессора).

Независимость программ от аппаратной части компьютера

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

Доступность всей оперативной памяти

В отличие от DOS средства управления оперативной памятью Windows обеспечивают доступность для программ всей оперативной памяти компьютера, а не только 640 Кбайт, что облегчает создание больших программ. Память компьютера с адресами больше 1024 Кбайт называется *расширенной* памятью. Непосредственный доступ к этой памяти возможен только в защищенном режиме работы процессора. Недостаток оперативной памяти решается в среде Windows с помощью *виртуальной памяти*. Виртуальная память – это расширение адресуемого пространства задачи, полученное за счет использования части внешней памяти. Эту часть памяти принято называть ***файлом подкачки***.

Распределение общей памяти

Операционная система распределяет память между задачами. В Windows 98 реализовано сегментно-страничное распределение памяти. Для каждой задачи выделяется сегмент памяти в ОЗУ. Этот сегмент в свою очередь делится на разделы фиксированного размера. При трансляции программ каждой задаче назначаются виртуальные адреса, начиная с нулевого, которые затем отражаются на реальные адреса оперативной памя-

ти. Все адресуемое пространство процессоров i386 равно 4 Гбайта. Если объем оперативной памяти недостаточен для выполнения задачи, то операционная система обращается к виртуальной памяти компьютера, то есть к жесткому диску.

При использовании виртуальной памяти также используется сегментно-страничный принцип распределения памяти. Виртуальное адресное пространство делится на страницы. Размер страниц выбирается кратным степени 2: 512 байт, 1024 байта и т.д. Все ОЗУ также делится на страницы того же размера. Операционная система создает для каждого процесса информационную структуру – таблицу страниц и устанавливает соответствие между номерами виртуальных и физических страниц, загруженных в ОЗУ, или делается отметка о том, что страница выгружена на диск.

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.

Объектно-ориентированная платформа операционной системы

Windows 98 - это объектно-ориентированная графическая операционная система. Все элементы операционной системы - это по сути объекты со своими свойствами. Вычисления в компьютере осуществляются путем обмена данными между объектами. Один объект требует, чтобы другой объект выполнил некоторые действия. Объекты взаимодействуют, посылая сообщения. *Сообщение* – это запрос на выполнение действия. Каждый объект имеет независимую память, которая состоит из других объектов. Каждый объект является представителем класса, который выражает свойства принадлежащих ему объектов. В классе задается поведение объекта, поэтому все объекты, принадлежащие к данному классу, могут выполнять одинаковые действия. Например, кнопка в текстовом процессоре, табличном процессоре или в системе управления базами данных выполняет одинаковые функции. Все классы образуют иерархическую древовидную структуру, отражающую иерархию наследования. Память и поведение, связанное с экземпляром определенного класса, могут использоваться любым классом, расположенным ниже в иерархической структуре.

Использование новых компьютерных технологий

Развитие графических операционных систем привело к возникновению понятия *компьютерные технологии* - это алгоритмы и их конкретная реализация для передачи, хранения и обработки информации в цифровом виде.

> *Компьютерные технологии - это алгоритмы и их конкретная реализация для передачи, хранения и обработки информации в цифровом виде.*

При разработке ОС фирмой Microsoft использованы различные технологии, обеспечивающие возможность повторного использования кода, упрощения взаимодействия приложений, позволяющие использовать программы, написанные на различных языках программирования, обмениваться данными между программами, обнаруживать новые программы и объекты, создавать пользовательские макропрограммы, окна диалога, меню и т. д. на базе компонентов, имеющихся в Windows, приложениях или загруженных из сети Internet.

Технология COM (модель компонентных элементов) - устанавливает открытый стандарт, который определяет принципы взаимодействия программ. COM – это архитектура программного обеспечения, позволяющая приложениям пользователя взаимодействовать с другими компонентами, в том числе поставляемыми другими разработчиками и содержащимися в коммерческих приложениях. COM – компоненты обычно представляют собой библиотеки классов - файлы с расширением *.exe, *.ocx, *.dll. В роли COM – компонентов могут выступать и такие приложения Windows, как Word, Excel, PowerPoint, Intrnet Explorer. В то же время стандартные приложения Windows (Word, Excel и др.) являются COM-серверами[4], поддерживающими ***Автоматизацию***.

Разработка программ на базе компонентов ускоряет программирование, позволяет собирать приложения из уже проверенных и стандартизированных компонентов, что значительно сокращает сроки разработки программных продуктов и повышает их качество. На основе COM – компонентов можно создавать ActiveX – компоненты.

ActiveX – это технология, основанная на COM, а ActiveX–компонент – это модуль исполняемого кода (например файлы с расширением .exe, .dll или .ocx), созданный в соответствии со спецификацией ActiveX на повторно используемые объекты. Несмотря на схожесть понятий, имеется существенная разница между объектно-ориентированным программированием (ООП) и разработкой программного обеспечения на основе ActiveX – технологии. ООП – это способ разработки программных компонентов на базе объектов только для одной среды разработки. А технология ActiveX – позволяет комбинировать объекты независимо от того, на каком языке они написаны. ActiveX и COM позволяют использовать компоненты, созданные на основе ООП во многих средах.

Автоматизация – это технология, позволяющая программам предоставлять свои объекты средствам разработки, макроязыкам и другим программам. Например, редактор электронной таблицы может предоставлять разработчику такие объекты, как таблица, ячейка, группа ячеек, а текстовый процессор - объекты в виде документа, абзаца, предложения, выделенного фрагмента. Любое клиентское приложение способно использовать внешний COM-компонент через автоматизацию, независимо от того, в каком файле он содержится - *.exe или *.dll.

Технология drag-and-drop позволяет перемещать объекты в окнах и на рабочем столе при помощи мыши.

Обмен 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.

В основе указанных механизмов обмена информацией лежит автоматизация.

Буфер обмена (ClipBoard)– 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. Буфер обмена – это буфер *межпрограммного обмена* данными. После того, как данные помещены в буфер, они могут быть вставлены из него в текущий или в любой другой документ, подготовленный данным приложением или другим приложением.

[4] Сервер – объект (программа), предоставляющий свои ресурсы другим объектам (программам). Клиент – объект (программа), запрашивающий ресурсы. Клиент-сервер – способ организации взаимодействия компонентов и программ. Клиент может инициировать выполнение процедур сервером и получать результаты.

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 – *OLE 1* – 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.

Усовершенствованная технология OLE – *OLE 2* позволяет редактировать внедренный объект в месте вставки, не передавая его в документ-родитель. В OLE 2 улучшена технология *drag and drop* за счет введения межоконного перемещения объектов, усовершенствовано связывание объектов. При перемещении объектов связь между ними не теряется, как было ранее, а отслеживается. Технологии DDE, OLE 1 и OLE 2 совместимы сверху вниз: приложение-сервер и приложение-клиент обмениваются по наиболее новой технологии, доступной им обоим. Если, например, приложение-сервер поддерживает технологию OLE 2, а приложение-клиент поддерживает технологию OLE 1, то активизировать вставленный объект в приложении-клиенте окажется невозможным.

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.

Установка оборудования

Установка оборудования основана на технологии *Plug and Play*. 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. Технология Plug and Play обеспечивает автоматическое распознавание устройств для их установки и настройки. Она обеспечивает динамический контроль состояния системы и автоматическое уведомление об этом программных приложений, обеспечивает интеграцию драйверов устройств, системных компонентов и пользовательского интерфейса. Windows обеспечивает динамическое изменение конфигурации системы, построенной на базе данной технологии. Эта технология позволяет работать также с устройствами, не подчиняющимися спецификации Plug and Play, упрощая их настройку и управление оборудованием.

Оборудование, выполненное в соответствии со стандартом Plug and 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 and 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.

Сетевые возможности

Операционная системы Windows обеспечивает пользователя большим комплексом услуг по работе во 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. Сетевые средства Windows обеспечивают включение компьютера во всемирную сеть Интернет (при наличии в компьютере сетевой карты) или в локальную одноранговую компьютерную сеть, или в локальную компьютерную сеть с выделенным сервером. В одноранговой компьютерной сети все компьютеры равноправны и могут выступать как в роли сервера, так и в роли клиента. В локальной компьютерной сети с выделенным сервером все управление работой сети, распределением ресурсов, обработку запросов других компьютеров-клиентов осуществляет один компьютер - сервер

Обеспечение бесперебойной и бесконфликтной работы внешних устройств

Так как к компьютеру может подключаться одновременно несколько внешних устройств, то приняты меры, исключающие конфликты между программами. Для этой цели используются *прерывания* и *каналы прямого доступа к памяти*.

Прерывания. В архитектуре компьютеров PC предусмотрено 15 аппаратных прерываний. Прерывание указывает компьютеру, что какой-то процесс требует его внимания. Прерывания передаются по специальным шинам IQR. Таких шин 15. Распределение номеров линий прерываний зависит от используемых системных шин. Большинство устройств не могут совместно использовать одинаковые прерывания. Если два устройства используют одинаковые номера прерываний, то это может привести к зависанию компьютера или к тому, что не произойдет никакой передачи сообщения. Нехватку номеров запросов прерываний решают в операционной системе Windows двумя способами:

- первый способ - создание нескольких аппаратных профилей. Каждый из этих профилей можно настроить на решение определенного типа задач. Выбрав при загрузке соответствующий профиль, можно получить доступ к тому или иному устройству;
- второй способ – использование шин последовательного доступа USB, IEEE 1394 или SCSI. Шина USB, как уже отмечалось, позволяет подключить последовательно 256 устройств, шина IEEE 1394 - 63, а SCSI – 7.

Каналы прямого доступа к памяти. Каналы прямого доступа к памяти (каналы DMA) предназначены для быстрой передачи данных к периферийному устройству. При передаче данных по каналу прямого доступа к памяти процессор не задействуется и может выполнять другие задачи. Обычный компьютер имеет 8 каналов прямого доступа, при этом каналы, назначенные для сетевых карт или контроллера жесткого диска, не могут использоваться другими устройствами, так как это может привести к потере важных данных. Некоторые устройства используют одновременно несколько каналов DMA, например адаптер Media Vision ProAudio Spectrum 16 использует два канала DMA. Каждому каналу прямого доступа выделяется определенная область оперативной памяти.

Распределение портов ввода-вывода. Чтобы программа, не перепутала данные, относящиеся к тому или иному процессу, каждому процессу назначается определенная область памяти – *порт*. В архитектуре PC порты ввода-вывода распределяются в памяти. Это позволяет микропроцессору получить к ним доступ, указав соответствующий адрес. Таким образом, каждое устройство, подключенное к порту ввода-вывода должно знать адрес этого порта, иначе информация попадет не по назначению.

Доступ к памяти компьютера в сети Интернет. Для доступа к определенной области памяти компьютера в сети Интернет, кроме адреса порта, необходимо знать IP-адрес компьютера (IP адрес см. раздел 7). Совокупность номера порта и IP-адреса на-

зывают ***сокет***. За некоторыми процессами номера портов закреплены. Так, например, порт 21 закреплен за службой удаленного доступа к файлам FTP, порт 23 – за службой управления telnet.

Адресация памяти. В памяти компьютера, в зависимости от способа адресации, различают две области (рис. 1.14): область ***непосредственной адресации,*** занимающую первые 1024 Кбайта с адресами от 0 до 1024 Кбайт - 1 и *расширенную память* с адресами большими 1024 Кбайт. Доступ к ячейкам этой памяти возможен при использовании специальных программ-драйверов или в защищенном режиме работы микропроцессора. Область памяти до 1 Мбайта, то есть область непосредственной адресации, тоже делится на две части: *стандартную область* - 640 Кбайт (64 Кбайта используются для служебных программ и данных операционной системы, остальные 576 Кбайт используются для программ и данных пользователя) и *верхняя память* - 384 Кбайта (256 Кбайт используется для области видеопамяти дисплея и служебных программ, а 128 Кбайт используются для программ начальной загрузки ОС (ПЗУ) и др.).

<table>
<tr><td colspan="4">Непосредственно адресуемая память 1 Мбайт</td><td>Расширенная память, свыше 1 Мбайт</td></tr>
<tr><td colspan="2">Стандартная память 640 Кбайт</td><td colspan="2">Верхняя память 384 Кбайта</td><td>64 Кбайта, область верхних адресов памяти</td></tr>
<tr><td>64 Кбайта, область системных программ и данных ОС</td><td>576 Кбайт, Область программ и данных пользователя</td><td>256 Кбайт, Область видеопамяти дисплея и служебных программ</td><td>128 Кбайт, область программ начальной загрузки и ОС и др.</td><td></td></tr>
</table>

Рис. 1.14. Логическая структура основной памяти компьютера

Стандартная область памяти используется различными устройствами для связи с процессором, для некоторых устройств базовые адреса памяти устанавливают с помощью переключателей или специальных программ- утилит. При выделении области памяти для сетевой карты указывается не только базовый (начальный) адрес, но и размер адресуемой памяти. При выделении адреса памяти для сетевой карты операционная система резервирует под него место. Эта область не будет использоваться процессором. Естественно, что при этом уменьшается объем системной памяти. Большинство более поздних ISA-карт могут использовать верхние адреса памяти. Многие устройства для шин VLB, PC, а также некоторые ISA-карты могут работать с областями памяти выше 1 Мбайт. Предпочтительны карты, работающие с верхними адресами памяти, так как их использование снижает вероятность конфликтов с операционной системой. Расширенная память используется главным образом для хранения данных и некоторых программ ОС, расширенную память часто используют для организации *виртуальных* (электронных) дисков. При этом небольшой объем памяти, до 64 Кбайт – высокая память, используется обычно для хранения программ и данных операционной системы.

Системный реестр

В операционной системе MS-DOS настройка операционной системы осуществляется с помощью файлов config.sys и autoexec.bat. В операционной системе Windows в этих файлах принципиальная необходимость отпала, хотя они по-прежнему могут присутствовать. Вместо этих двух небольших файлов в ОС Windows для хранения сведений об операционной системе, установленном оборудовании и настройках используется большая база данных, называемая *системный реестр*. 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 реестра осуществляется операционной системой автоматически. Однако возможна и ручная настройка системного реестра. Организация системного реестра напоминает структуру жесткого диска: реестр имеет несколько разделов, а его иерархическая структура ничем не отличается от структуры жесткого диска с папками и файлами. Разделы реестра называют ***ветвями***. В каждой ветви имеются разделы. Разделы, в свою очередь, могут содержать подразделы и *параметры*, в которых хранится такая информация, как числа и текст. Существует две ветви верхнего уровня. Первая ветвь (HKEY_LOCAL_MACHINE) содержит всю информацию об аппаратном обеспечении компьютера (драйверы всех подключенных устройств и параметры их работы), информация относительно установки программного обеспечения, общая для всех пользователей. Во второй ветви (HKEY_USERS) находятся подразделы, соответствующие отдельным пользователям и содержащие информацию об их индивидуальных настройках (цвет рабочего стола, заставки, звуковые эффекты и др.).

Кроме этого, имеется еще три виртуальных раздела:

- раздел HKEY_CURRENT_USER является подразделом раздела HKEY_USERS и содержит информацию о пользователе, работающем в данный момент на компьютере;
- раздел HKEY_CURRENT_CONFIG содержит информацию о текущей аппаратной конфигурации;
- раздел HKEY_CLASSES_ROOT хранит сведения о соответствии между расширениями имен файлов и приложениях, с помощью которых они запускаются, а также соответствия между приложениями и ярлыками.

Реестр можно редактировать, но использовать эту возможность рекомендуется только *опытным пользователям*, вмешательство в настройку реестра неопытного пользователя может привести к фатальным последствиям - операционная система не будет загружаться.

Наличие средств мультимедиа

Операционная система Windows обеспечивает интерактивную работу с высококачественным звуком и видео при помощи специальных аппаратных и программных средств. В группу мультимедийных программ входят четыре стандартные программы Windows 98: Звукозались, Лазерный проигрыватель, Универсальный проигрыватель и Регулятор громкости.

Средства автоматизации

Средства автоматизации позволяют автоматически запускать программы при включении компьютера, настраивать режим копирования файлов на нужное устройство, настраивать главное меню программы, автоматически рассылать документы по электронной почте, принимать и обновлять файлы с каналов Интернет, на которые оформлена подписка, осуществлять дистанционное техническое обслуживание компьютера в случае неполадок в его работе, автоматическое обслуживание операционной системы (дефрагментация жесткого диска, проверка диска, очистка жесткого диска, запуск программ по расписанию).

1.5.2. ОСНОВЫ РАБОТЫ В WINDOWS

Рабочий стол

Запуск операционной системы и выключение компьютера

После включения питания программа проверяет исправность оперативной памяти и наличие подключенных внешних устройств и затем осуществляется загрузка операционной системы в оперативную память компьютера. Если на компьютере установлено несколько операционных систем, то на экран выводится список установленных операционных систем и пользователю предлагается выбрать нужную операционную систему.

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

После загрузки операционной системы на экране появляется рабочий стол программы Windows (рис. 1.15).

Для выключения компьютера следует выбрать команду ***Пуск*** и ***Завершение работы***. Операционная система сохраняет установленную конфигурацию в системном регистре и завершает работу. При этом питание компьютера выключается автоматически. 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.

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 удаленный файл может быть восстановлен. 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.*

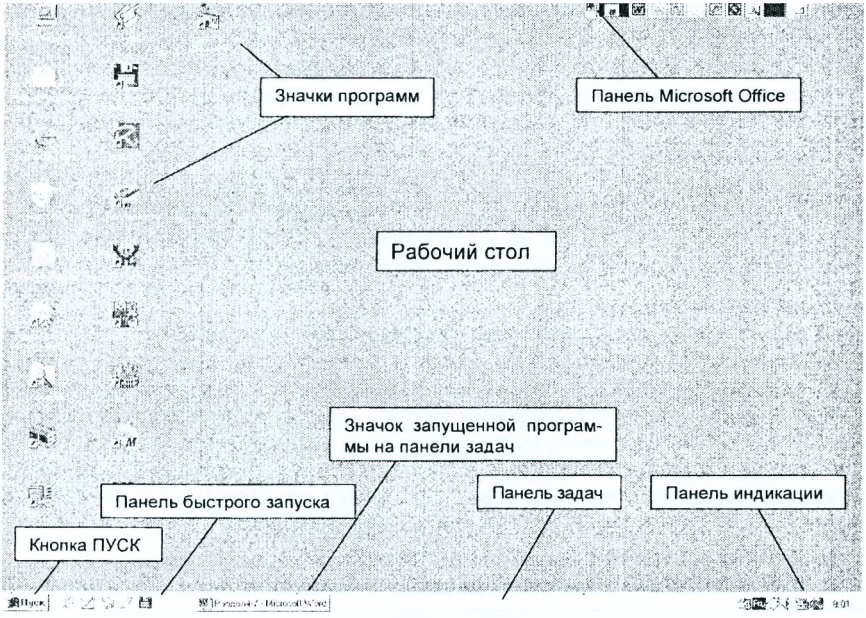


Рис.1.15. Рабочий стол

Кнопка ПУСК открывает доступ к командам *Главного меню*. Чтобы изучить возмож-

ности 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 появляется всплывающая подсказка о назначении соответствующего значка или экранной кнопки (в дальнейшем изложении экранные кнопки будем называть просто - кнопки).

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 Интернет, 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 Интернет, к которым приходится обращаться наиболее часто, чтобы они были всегда под рукой.

На панель задач выводятся значки всех запущенных программ. Поэтому легко перейти от одной программы к другой щелчком мыши по соответствующему значку.

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, а также узнать свойства объекта.

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 с панели 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.

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, то 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 закрашивается, 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 тем, что при ее отпускании в новом положении объекта появляется меню, которое предлагает выбрать операцию копирования или перемещения.

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 свойства. 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.

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.

На рис.1.16 представлено окно программы Проводник. В верхней части окна размещается строка заголовка (2), в которую выводится наименование открытой папки. В левой части строки заголовка размещается кнопка системного меню (1), а справа - кнопки свертывания (6), развертывания (7) и закрытия окна (8). Если окно развернуто, то на месте кнопки развертывания окна появится кнопка восстановления предыдущего со-

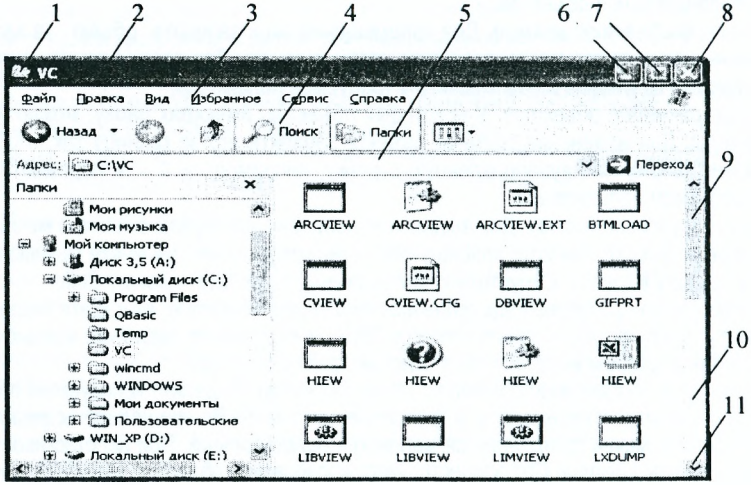


Рис. 1.16. Окно программы Проводник ОС Windows XP

стояния окна. Ниже строки заголовка размещается строка главного меню (3). При выборе пункта главного меню открывается всплывающее меню второго уровня. Меню второ-

го 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 Линейки прокрутки появляются в окне программы в том случае, когда информация не умещается в открытой области окна. Линейки прокрутки могут быть горизонтальные и вертикальные. На них размещаются элементы управления: 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. 1.17. приведены основные элементы управления, используемые в окнах Windows. К таким элементам относятся:

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

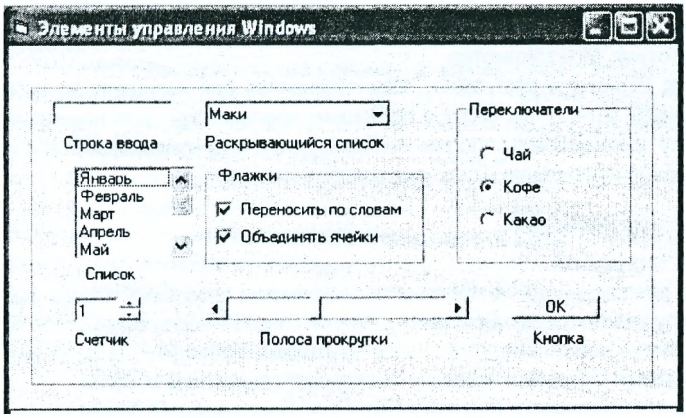


Рис. 1.17. Элементы управления Windows

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.

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 Панель Microsoft Office. Чтобы поместить значок программы на рабочий стол, создайте для нее ярлык и переместите его на рабочий стол.

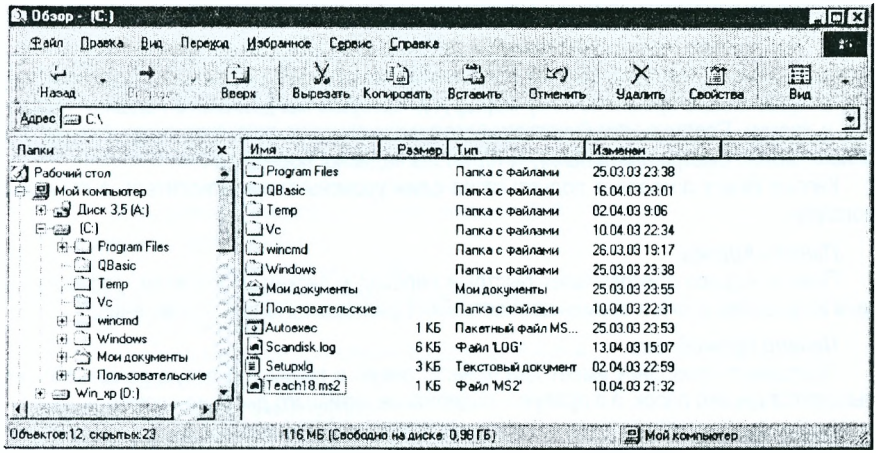


Рис. 1.18. Программа Проводник ОС Windows 98

Окно программы

Окно программы Проводник является типичным окном Windows (рис. 1.18). В верхней части окна выводится строка меню. Состав пунктов меню зависит от операционной системы (см. рис. 1.16, на котором приведено окно программы 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.

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 Интернете.

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.

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 значки, определяющие тип файла, и их имена. При выводе файлов в панель в виде *таблицы* на экран выводятся дополнительно размер, тип файла, дата и время его создания или последнего обновления.

Полные сведения о каждой папке и файле можно получить с помощью команды *Свойства* 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, а лучше создайте ярлык для данного файла и поместите его в нужную папку.*

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.

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;

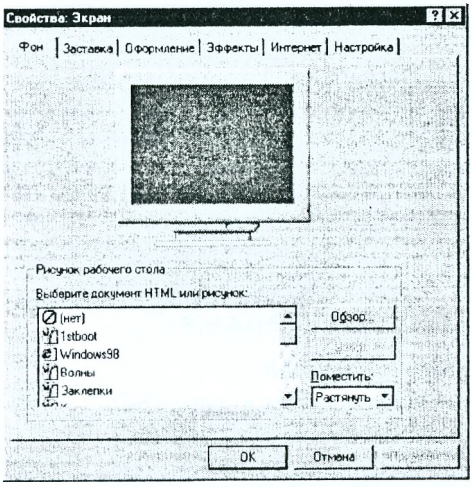


Рис. 1.19. Настройка рабочего стола

- введите команду ***Рабочий стол Active Desktop, Настройка рабочего стола.*** Появится окно диалога для настройки параметров рабочего стола (рис.1.19). 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 открытием 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"). *Не рекомендуется* изменять настройки, установленные по умолчанию.

Настройка раскладки клавиатуры

Настройка раскладки клавиатуры позволяет установить время задержки перед началом повторения (закладка "Скорость") и скорость повтора символов при удержании нажатой клавиши на клавиатуре, 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. 1.20) необходимо ввести команду ***Пуск, Настройка, Панель управления, Клавиатура***.

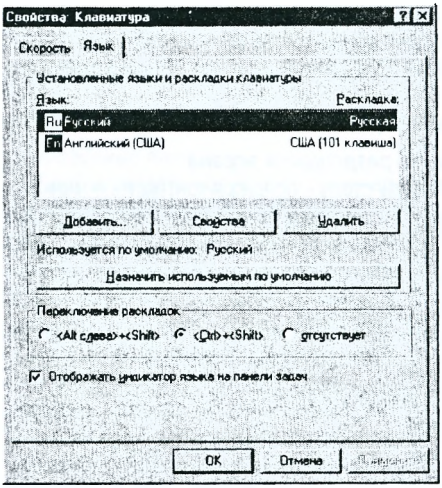


Рис. 1.20. Настройка клавиатуры

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 индикации 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.

1.5.3. СЕРВИСНАЯ ОБОЛОЧКА WINDOWS COMMANDER

Для работы с программами, разработанными для работы в среде MS-DOS, используются программы Windows Commander, Total Commander, Far Manager и др. Одной из причин необходимости использования этих сервисных оболочек является то, что не все досовские программы запускаются из среды Windows. Другой не менее важной причи-

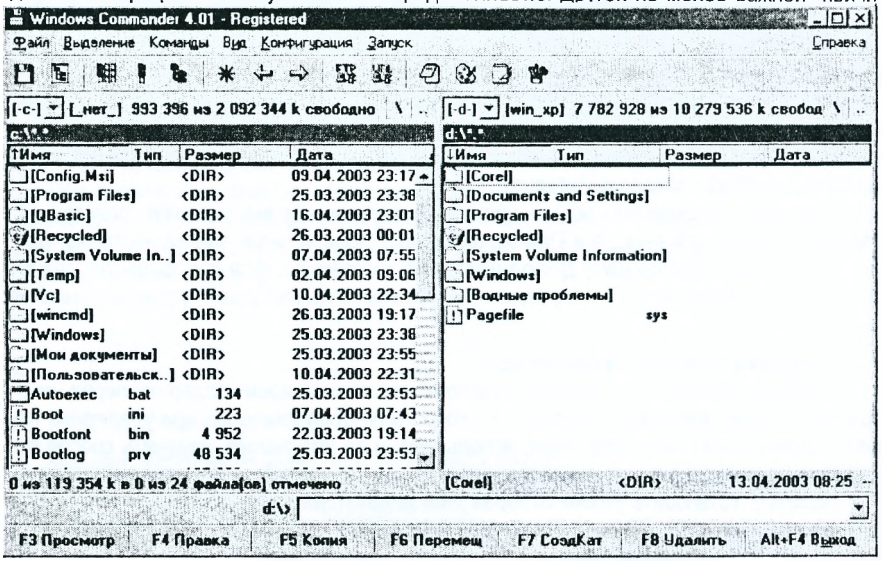


Рис. 1.21. Сервисная оболочка Windows Commander

ной является то, что многие пользователи привыкли работать с сервисными оболочками операционной системы MS-DOS и хотят видеть похожий интерфейс и в среде Windows. К достоинствам Windows Commander относится во-первых, то, что ее интерфейс похож на интерфейс уже упоминавшейся программы Far и других популярных сервисных оболочек для работы с командами операционной системы MS-DOS, например Volcov Commander, Norton Commander. 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. 1.21. 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, в которую выведено назначение функциональных клавиш.

Главное меню

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, по-прежнему 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. 1.22). 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.

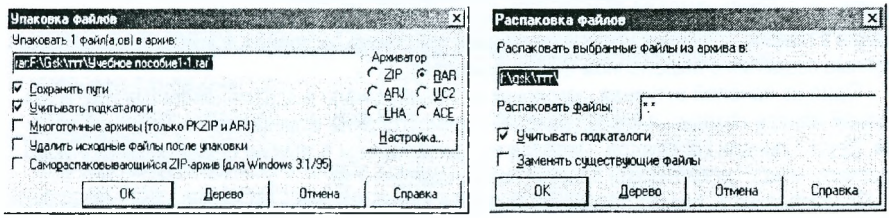


Рис. 1.22. Упаковка и распаковка файлов

Команда *Разбить* позволяет разбить длинный файл на части (рис. 1.23). 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.*

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.

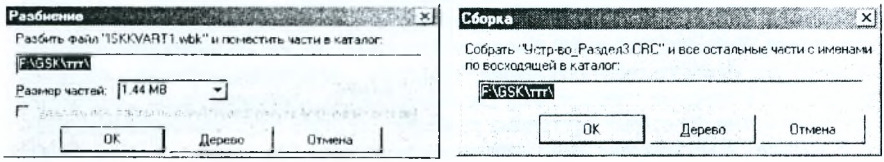


Рис. 1.23. Разбивка и сборка файла

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*.

Меню ***Запуск*** позволяет производить настройку меню запуска. 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.

Для любителей работать с клавишами следует помнить, что для большинства команд зарезервированы комбинации клавиш, которые указаны в соответствующем меню.

Панели

Панели предназначены для вывода списков папок и файлов на дисках. В заголовке панели указано имя диска, папки 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. В активной панели находится курсор панели – прямоугольник, отмеченный пунктирной линией. Другая панель нетекущая. Переход из одной панели в другую осуществляется либо щелчком мыши по нетекущей панели, либо клавишей Tab.

Информация в панели может выводиться в полной или краткой форме. В краткой форме на экран выводится только имя и тип файла (расширение имени файла), в полной форме на экран дополнительно выводятся размер, дата и время создания, а также атрибуты файла.

В первых строках панели выводятся имена папок диска. Папки отмечаются значками в виде желтого прямоугольника, справа от имени каталога изображается <DIR>. Самую верхнюю строку занимает ссылка на родительский каталог. Для корневого каталога ссылки на родительский каталог нет. В поле имени для родительского каталога изображается стрелка с двоеточием - символ "..", а справа от него <DIR>.

Имена файлов выводятся строчными буквами. Для файлов с атрибутами "скрытый"

или "системный" на значке файла отображается восклицательный знак красного цвета.

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.

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 двойным щелчком мыши. При необходимости указать при вводе команды дополнительные параметры необходимо воспользоваться строкой ввода: запишите команду (или имя файла) в строку ввода, допишите необходимые параметры и нажмите Enter.

Ввод команд главного меню осуществляется щелчком мыши. Для ввода команд главного меню можно использовать также *горячие клавиши* - символы, выделенные в команде символом подчеркивания, в комбинации с клавишей Alt. При управлении с помощью клавиатуры для перехода в главное меню нажмите клавишу Alt. Для перемещения по командам меню используются клавиши управления перемещением курсора. Для вывода меню второго уровня нажмите клавишу Enter или Пробел. Для выхода из меню нажмите клавишу Esc.

Выделение файлов

Все операции в среде Windows Commander выполняются над выделенным файлом или папкой. Для выделения файла (папки) щелкните по нему мышью. Выделение группы файлов выполняется так же, как и в программе Проводник с использованием клавиш *Shift* и *Ctrl*. Для выделения группы файлов можно использовать также клавишу *Insert*: установите курсор на имя файла и нажмите клавишу Insert – файл выделяется красным цветом. Повторное нажатие клавиши Insert на выделенном файле отменяет выделение. Для группового выделения однотипных файлов используется маска. Чтобы выделить файлы по маске, выберите команду ***Выделение, Выделить группу,*** укажите в окне диалога маску или выберите тип файлов по шаблону и щелкните по кнопке ОК. Для отмены выделения введите команду ***Выделение, Снять выделение***.

Работа с файлами

Файлы можно создавать, просматривать, копировать, перемещать, переименовывать и удалять.

Для создания небольшого файла можно воспользоваться блокнотом NotePad или редактором Write, значки которых находятся в панели инструментов.

Для просмотра текстовых файлов выделите нужный файл и нажмите клавишу F3. По умолчанию просматривать можно только текстовые файлы написанные с использованием кодовой таблицы ASCII. Параметры просмотра/правки можно изменить командой ***Конфигурация, Настройка***, закладка ***Правка/ Просмотр***.

Правка выделенных файлов осуществляется нажатием клавиши F4. По умолчанию для правки файлов используется редактор NotePad. Имеется возможность изменить настройку и указать другой редактор для просмотра файлов, как указано выше.

Для копирования файлов можно использовать технологию перетаскивания файлов:

1. Выведите в одну из панелей, например левую, диск и папку, где находится копируемый файл;

2. Выведите в другую, нетекущую панель оглавление папки-назначения;

3. Зацепите копируемый файл (файлы) в левой панели и переместите его в правую панель.

Аналогично выполняется копирование с использованием клавиши F5. После выполнения подготовительных операций по п. п. 1, 2 , выделите копируемые файлы, нажмите клавишу F5, а затем Enter. По умолчанию файл копируется в нетекущую панель. При необходимости можно изменить маршрут, используя кнопку Дерево окна диалога.

Клавиша F6 позволяет выполнить переименование или пересылку файлов. При переименовании в окне диалога необходимо указать только новое имя файла, так как файл остается на своем месте. При пересылке указывается маршрут и новое имя файла. Операция пересылки выполняется так же как и операция копирования, но после ее выполнения исходный файл удаляется с диска.

Для удаления файла выделите его и нажмите клавишу F8.

Управление папками (каталогами)

Для открытия папки (входа в каталог) щелкните по ней дважды мышью или нажмите клавишу Enter. Для выхода из папки (каталога) щелкните дважды мышью по имени родительского каталога или установите курсор на имя родительского каталога и нажмите клавишу Enter. Для создания папки на текущем диске нажмите клавишу F7 и введите в окне диалога имя папки. Для переименования папки выделите ее курсором, нажмите клавишу F6 и введите в окне диалога новое имя. Для удаления папки выделите ее курсором и нажмите клавишу F8. На экран будет выведено окно с предупреждением. Можно отказаться от удаления или подтвердить удаление папки.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какая информация о файле выводится при полной и краткой форме представления каталога?
2. Какая панель является текущей?
3. Как перевести курсор в другую панель?
4. Что означает выражение "выделить файл", "выделить каталог"?
5. Как "войти" в каталог? Как "выйти" из каталога?
6. Поясните порядок выполнения команд из среды Windows Commander.
7. Поясните порядок запуска исполняемых и командных файлов.
8. Выведите для редактирования файл autoexec.bat.
9. Выведите в правую панель оглавление диска A:.
10. Выведите в левую панель каталог в полной форме по алфавиту, по расширению имени файла, по дате изменения.
11. Выведите в правую панель каталог в краткой форме.
12. Как осуществляется архивация файлов в Windows Commander?
13. Как установить атрибуты файлов в Windows Commander?
14. Как просмотреть скрытые или системные файлы в Windows Commander?

ЗАКЛЮЧЕНИЕ

Операционная система Windows 98 является одной из наиболее популярных операционных систем для персональных компьютеров фирмы Intel и рабочих станций в локальных сетях ЭВМ. Несмотря на появление новых операционных систем она сохраняет свое значение и часто устанавливается вместе с операционной системой Windows XP для обеспечения программной совместимости с программами написанными для операционной системы MS-DOS. Переход от работы в Windows 98 к работе в среде Windows XP не вызывает у пользователей серьезных затруднений.

1.6. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О РАБОТЕ НА ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ

1.6.1. АРХИВАЦИЯ ФАЙЛОВ

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. Эти файлы могут пригодиться для восстановления операционной системы. Для создания загрузочного диска введите команду ***Пуск, Настройка, Панель управления, Установка и удаление программ*** и откройте закладку ***Загрузочный диск.***

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* (или упаковкой), процесс восстановления файла в первоначальном виде – *разархивированием* (или восстановлением). Упакованный (сжатый) файл принято называть *архивом*.

Архивация 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 $К_с$, 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 разные методы: метод Хаффмана, RLE-кодирование, LZW-кодирование.

Метод Хаффмана основан на том, что частота повторения символов в тексте различная. Поэтому можно заменить символы вспомогательным кодом. Чем больше часто-

та повторения символов, тем меньше должна быть длина кода. Кодовая таблица, образовавшаяся при архивации, хранится вместе с архивным файлом.

RLE-кодировка основана на замене повторяющихся последовательностей байтов одной последовательностью с указанием числа повторений.

LZW-кодирование – словарный метод, наиболее распространенный метод архивации. Используется словарь, состоящий из последовательностей данных или слов. При сжатии эти слова заменяются на их коды из словаря. В наиболее распространенном варианте реализации в качестве словаря выступает сам исходный блок данных. Основным параметром словарного метода является размер словаря. Чем больше словарь, тем больше эффективность. Для эффективной работы данного метода при сжатии требуется дополнительная память, приблизительно на порядок больше, чем нужно для исходных данных словаря. Существенным преимуществом словарного метода является простая и быстрая процедура распаковки. Дополнительная память при этом не требуется. Такая особенность особенно важна, если необходим оперативный доступ к данным.

Программы для архивации файлов

Существует много программ для архивации файлов. Как правило, эти программы позволяют помещать копии файлов в архивный файл на диск в сжатом виде, 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. В операционной системе Windows наиболее часто применяются программы WinZip и WinRar. Программа WinZip обладает высокой скоростью сжатия, но степень сжатия достаточно низкая. WinRar обеспечивает большую степень сжатия, но проигрывает WinZip в скорости сжатия данных. В операционной системе MS-DOS широко распространены программы ZIP, ARJ, ICE, LHA, RAR.

Все без исключения современные архиваторы используют один и тот же алгоритм сжатия без потери информации – словарный LZW.

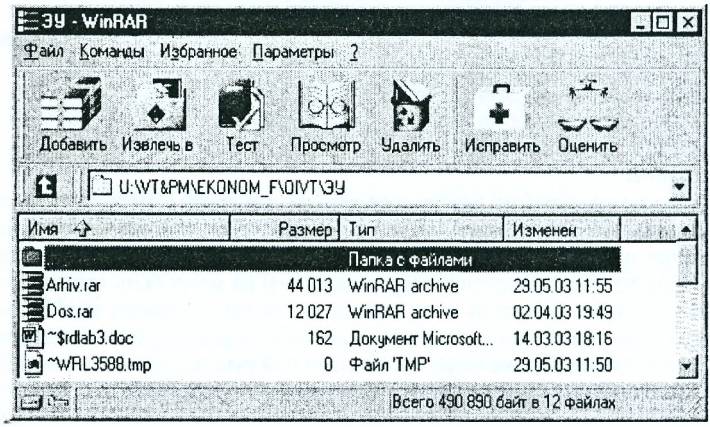


Рис. 1.24. Архиватор WinRar

Программа архивации WinRAR

Для архивации файлов в среде Windows разработаны две программы WinZip и WinRAR. Программа WinRAR имеет более удобный интерфейс (рис. 1.24). Алгоритм работы при архивации и разархивации файлов также предельно прост.

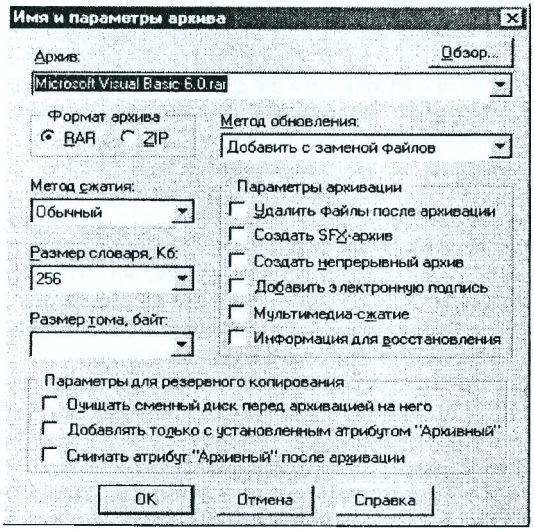


Рис.1.25. Настройка параметров архива

Алгоритм архивации файлов:

– выделите в окне архивируемый файл (папку), используя раскрывающийся список адресов;

– щелкните по кнопке *Добавить* – открывается окно диалога (рис.1.25);

– установите требуемые атрибуты архивации, метод обновления и укажите маршрут, куда следует поместить архивный файл, используя строку ввода или кнопку *Обзор*. Имя архивного файла можно принять по умолчанию;

– после настройки параметров архива и маршрута щелкните по клавише *ОК*.

Алгоритм извлечения файлов

Алгоритм извлечения файлов из архива также прост:

– выделите в окне диалога (рис. 1.24) архивный файл;

– щелкните 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.

1.6.2. ЗАЩИТА ОТ КОМПЬЮТЕРНОГО ВИРУСА

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.

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, файлово-загрузочные;

– *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.

Профилактика против заражения вирусом

Для предотвращения заражения ПЭВМ компьютерным вирусом, а также облегчения процедуры "лечения" зараженного 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 только защищенной от записи эталонной дискеты с операционной системой. Во избежании заражения вирусом, распространяющимся через загрузочные секторы дискет, перед перезагрузкой компьютера с жесткого диска убедитесь в отсутствии дискеты в дисководе А:;

– 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 установить на компьютер антивирусную программу Касперского (AVP). После установки она будет запускаться при каждой загрузке компьютера. Установка программы несколько замедлит работу компьютера;

– проверка чужой дискеты на отсутствие вируса перед ее загрузкой в компьютер.

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 специфическая 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. Программа-доктор, ориентированная на некоторый фиксированный набор вирусов, не в состоянии будет "вылечить" файлы, зараженные другими типами вирусов, кроме того программы-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 дискет.

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.

Действия при заражении компьютера вирусом

При заражении компьютера вирусом или подозрении на заражение вирусом рекомендуется следующий порядок действий:

1. Запустить программу AVP – монитор для проверки и уничтожения вирусов на дисках.

2. После обнаружения вируса следует последовательно обезвредить все дискеты, которые могли подвергнуться заражению вирусом. Если на дискете нет нужных файлов, копий которых нет в архиве, то рекомендуется заново отформатировать дискету, а затем восстановить файлы с архивных дисков.

При отсутствии опыта борьбы с вирусами рекомендуется обратиться к специалисту. И не забудьте оповестить об обнаруженном вирусе всех товарищей, с которыми вы обмениваетесь программами, о возможности заражения их компьютеров вирусами.

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?

ЗАКЛЮЧЕНИЕ

Программы архивации обеспечивают сжатие данных при сохранении резервных копий программ и данных на внешних носителях информации и восстановление их, в случае необходимости, в первоначальном виде.

Антивирусные программы обеспечивают защиту компьютера от программных вирусов, распространяющимися через загрузочные сектора дисков, исполняемые файлы или сети ЭВМ. Для обеспечения защиты компьютера от заражения вирусами необходимо соблюдать несложные правила и установить на компьютер одну из популярных антивирусных программ.

2. МАТЕМАТИЧЕСКАЯ СИСТЕМА MATHCAD

2.1. ОБЩИЕ СВЕДЕНИЯ

2.1.1. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

Математической система MathCad является универсальной математической системой, ориентированной на проведение математических и инженерно-технических расчетов. MathCad позволяет:

- выполнять вычисления различной степени сложности, как в символьном виде, так и в числовой форме;
- отображать информацию в графическом виде, что облегчает визуализацию и анализ данных;
- описывать алгоритмы решаемых задач с помощью встроенного языка программирования;
- оформлять результаты решения задач в привычной математической форме, благодаря встроенному текстовому и графическому редакторам;
- получать разнообразную справочную информацию посредством справочной системы и "Центра ресурсов", включающих множество практических примеров;
- осуществлять обмен данными с другими WINDOWS – приложениями;
- автоматически обновлять результаты расчетов и графики при изменении исходных данных;
- использовать привычный способ математической записи условий задачи и задания исходных данных, принятый в литературе.

Кроме того, имеется большое количество научной и учебной литературы, облегчающей изучение математической системы MathCad.

2.1.2. ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

Интерфейс пользователя математической системы MathCad (рис.2.1.) включает: главное меню; стандартные панели инструментов и панели инструментов семейства

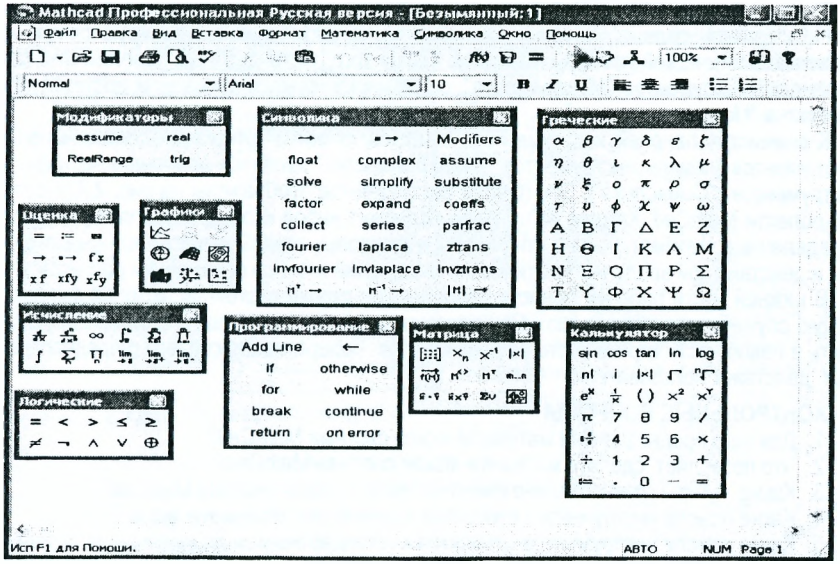


Рис.2.1. Интерфейс пользователя математической системы Mathcad

"Математические"; рабочую область и строку состояния.

Главное меню состоит из девяти пунктов, в каждом из которых объединены функционально однородные команды:

- ***Файл (File).*** Пункты данного меню отражают общие действия по работе с документами, такие как создание, открытие, сохранение и печать документов с результатами расчетов.

-***Правка (Edit).*** В данном меню сосредоточены общие команды по редактированию документа или его фрагментов, такие как: вырезать; копировать; вставить; проверить орфографию; отменить ранее выполненное действие; найти заданный фрагмент и, если необходимо, заменить его на другой.

- ***Вид (View).*** Определяет вид интерфейса пользователя, то есть перечень и порядок расположения различных элементов, отображаемых на экране дисплея при работе в MathCad. Данное меню управляет масштабом, отображением линеек, строкой состояния, набором активных панелей инструментов.

- ***Вставка (Insert).*** Позволяет ввести в документ различные элементы: матрицы; графики; функции; рисунки и компоненты других программных приложений, например, Excel, MatLab.

- ***Формат (Format).*** Задает формат стиля, шрифта текста, абзаца. Позволяет установить вид формул и формат отображения результатов расчета.

- ***Математика (Math).*** Содержит команды, обеспечивающие проведение расчетов и задание опций, определяющих их точность.

- ***Символика (Symbolics).*** Включает команды выполнения различных символьных расчетов.

- ***Окно (Window).*** Команды данного меню активизируют различные документы и определяют режимы их отображения на экране дисплея.

- ***Помощь (Help).*** Команды данного меню позволяют получить инструкции по работе с математической системой MathCad, а также вызвать *"Центр ресурсов"*, содержащий многочисленные примеры решения задач

К стандартным панелям инструментов относятся панели инструментов *"Стандартная"* и *"Форматирование"*, которые, как правило, по умолчанию активизируются при загрузке математической системы MathCad. Кроме того, к стандартным панелям относится панель инструментов *"Математика"*, содержащая ссылки на панели инструментов семейства *"Математические"*.

К семейству панелей инструментов *"Математические"* относятся девять панелей инструментов (панель инструментов *"Модификаторы"* является дополнением панели инструментов *"Символика"*). Эти панели инструментов отображены на рис. 2.2. в рабочей области MathCad. Каждая из панелей содержит набор кнопок, позволяющих ввести определенные математические операторы и символы, задать вычислительные операции и действия при вводе математических выражений или при проведении расчетов.

В нижней части рабочей области расположена строка состояния, которая содержит общую служебную информацию. При перемещении курсора мыши в определенную область в левой части строки состояния выводится оперативная справка о предполагаемых действиях пользователя.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначена математическая система MathCad?
2. Что позволяет сделать математическая система MathCad?
3. Какие пункты главного меню имеет математическая система MathCad?
4. Какие панели инструментов относятся к семейству "Математические"?
5. Какие панели инструментов относятся к стандартным и по умолчанию активизируются при загрузке математической системы MathCad?

2.2. ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ И ТИПЫ ДАННЫХ

2.2.1. ПЕРЕМЕННЫЕ И ФУНКЦИИ, ВЫЧИСЛЕНИЕ ПРОСТЕЙШИХ ВЫРАЖЕНИЙ И ФУНКЦИЙ

Переменные. Длина имени переменной не ограничена. В имени переменной допускается использовать все символы латинского языка, числа, греческие символы и ряд специальных символов: штрих, бесконечность, символ подчеркивания, знак процента, нижний индекс.

Редактор MathCAD различает прописные и строчные буквы, а также их начертание. Например, символы x и *x* или x и X для редактора будут разными символами. При написании имен переменных необходимо соблюдать некоторые требования:

- имя переменной не может начинаться с цифры;
- символ ∞ может быть только первым символом в имени переменной;
- все символы в имени должны иметь один стиль и шрифт;
- не допускается использование в качестве имен переменных зарезервированные слова.

Допускается использовать в качестве имен переменных выражения, например: [a+b] или a+b.

Для ввода нижнего индекса в имени переменной, например A_i, следует ввести символ "[". Для выхода из режима ввода нижних индексов необходимо нажать клавишу "Пробел".

Чтобы определить значение переменной используется символ присваивания ***":="***, который вводится с помощью горячих клавиш ***SHIFT +[:]*** или нажатием соответствующей кнопки на панели инструментов *"КАЛЬКУЛЯТОР"* и *"ОЦЕНКА"*. Чтобы отобразить значение переменной применяется знак равенства **"="**, который задается посредством горячей клавиши ***[=]*** или нажатием соответствующей кнопки на панели инструментов *"КАЛЬКУЛЯТОР"* и *"ОЦЕНКА"*. Пример.

$$r := 12 \qquad s := \pi \cdot r^2 \qquad s = 452.389$$

Функции. В MathCad входят функции двух типов: встроенные функции и функции пользователя. Количество встроенных функций MathCad около 300. Их можно ввести с клавиатуры или с помощью мастера функций, который активизируется с помощью горячих клавиш ***CTRL+E***, пиктографического значка ***f(x)*** или меню ***Вставка, Функция***. При этом отображается диалоговое окно "Вставить функцию" (рис. 2.2), в котором выбирается функция и задаются ее аргументы. Некоторые функции можно ввести также используя операторы панелей инструментов "Математика".

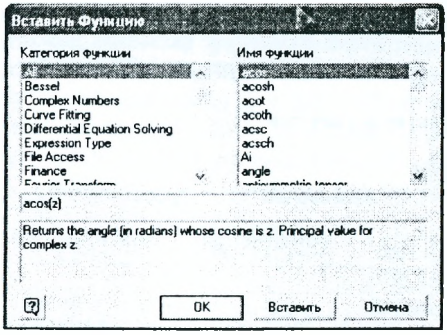


Рис.2.2. Окно "Вставить функцию"

Для функции пользователя задается ее имя, после чего в скобках через запятую указывается перечень аргументов функции. Далее вводится символ присваивания, после которого задается выражение, определяющее вид самой функции.

Если значение аргумента тригонометрической функции (*sin, cos, tg*) задается в градусах, необходимо перевести его в радианы. Это достигается посредством функции ***deg.***

Примеры:

$skr(r) := \pi \cdot r^2$ $skr(12) = 452.389$ $t := 4$ $skr(t) = 50.265$

$a := 1.2$ $b := 0.4$ $\sin(a)^2 + \cos(b^2) = 1.856$

$a := 90$ $\sin(a) = 0.894$ но $\sin(\deg a) = 1$

При организации вычислений в MathCad необходимо учитывать следующие особенности. Выполнение расчетов в MathCad происходит слева направо и сверху вниз. Данную последовательность следует учитывать при вводе алгоритма задачи и строго ей следовать во избежание ошибок и получения неверных результатов.

2.2.2. ДЕЙСТВИТЕЛЬНЫЕ И КОМПЛЕКСНЫЕ ЧИСЛА

Все действительные числа в Mathcad хранятся как числа двойной точности с плавающей точкой, которые при вводе и выводе могут отображаться:

- в формате целых чисел, например, 345;
- в виде десятичной дроби (в качестве разделителя используется точка), например, 2.456;

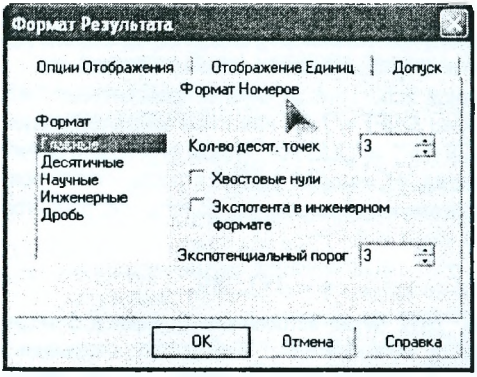


Рис. 2.3. Окно "Формат результата"

- в виде простой дроби, например, $\frac{1}{3}$;
- в виде смешанной дроби, например, $4\frac{3}{5}$;
- в экспоненциальной форме, например, $2.654*10^{32}$.

Можно указать формат отображения результатов расчетов. Это обеспечивается посредством меню ***Формат, Результат***. При этом появляется диалоговое окно (рис.2.3.), в котором указывается формат отображения результатов. Пример использования различных форматов для отображения результатов расчетов представлен на Листинге 2.1.

Листинг 2.1. **Форматы представления результатов расчетов**

Форматы				
Главный	Десятичный	Научный	Инженерный	Дробь
$\frac{8}{7} = 1.14286$	$\frac{8}{7} = 1.1429$	$\frac{8}{7} = 1.143E{+}000$	$\frac{8}{7} = 1.143 \times 10^{0}$	$\frac{8}{7} = 1\frac{1}{7}$

При работе с комплексными числами следует учитывать некоторые особенности организации вычислений в MathCad:

- для многозначной функции MathCad возвращает значение, составляющее на комплексной плоскости минимальный положительный угол с положительным направлением действительной оси ,например, $\sqrt[4]{-16} = 1.414 + 1.414i$;

- следует помнить специфику выполнения арифметических операций в MathCad, идентичных с точки зрения классической математики, но различающихся в написании. Подкоренное выражение рассматривается как действительное, а степень числа - как комплексное, например:

$$\sqrt[3]{-1} = -1 \quad \blacksquare, \text{ но } (-1)^{1/3} = 0{,}5+0{,}866i$$

В MathCad существует набор встроенных функций для работы с комплексными числами, которые приведены в таблице 2.1.

Таблица 2.1. Встроенные функции для работы с комплексными числами

Обозначение в Mathcad	*Описание*	*Пример Z:=2.34+3.14i*
Im(z)	Вычисление мнимой части комплексного числа	Im(z) = 3.14
Re(z)	Вычисление действительной части комплексного числа	Re(z) = 2.34
arg(z)	Определение значение угла, между радиусом – вектором и осью X	arg(z) = 0.93
csgn(z)	Принимает значение=0, если Z=0; значение =1,если Re(z)>0 и Im(z)>0; значение -1 в других случаях	csgn(z) = 1
signum(z)	Принимает значение 0, если Z=0; Z/|Z| - в других случаях.	signum(z) = 0.598 + 0.802i

Кроме этого, при работе с комплексными числами следует учитывать, что |Z| определяет модуль комплексного числа, а $\overline{Z}$ - число, комплексно сопряженное с Z. Например:

$$Z := 2.34 + 3.14i \quad |Z| = 3.916 \quad \overline{Z} = 2.34 - 3.14i$$

Чтобы ввести $\overline{Z}$, необходимо задать Z, а затем нажать клавишу ["].

Математические константы

В математической системе MathCad используются общепринятые математические константы. Они отображаются так же как и в классической математике, поэтому их использование и восприятие при вычислениях затруднений не вызывает. Математические константы MathCad представлены в таблице 2.2.

По умолчанию математические константы в вычислениях учитываются с точностью до 15-и значащих цифр. Для того, чтобы отобразить значения математических констант с максимальной точностью, необходимо задать требуемую точность отображения данных и результатов вычислений. Для этого необходимо воспользоваться меню ***Формат,***

Таблица 2.2 Математические константы MathCad

Наименование	*Обозначение*	*«Горячие» клавиши*	*Точность представления*
Пи	π	*CTRL +SHIFT+[p]*	До 15-и значащих цифр
Основание натурального логарифма	e	*Клавиша [e]*	До 15-и значащих цифр
Мнимая единица $\sqrt{-1}$	*i или j*	*Клавиша [i] или [j]*	
Бесконечность	∞	*CTRL+SHIFT+[z]*	10^{307}
Процент	%	*Клавиша [%]*	0.01

Результат и в диалоговом окне *"Формат Результата"* на вкладке *"Формат Номеров"* установить требуемое значение *"Кол-во десят. точек"* (рис. 2.3). В результате данного дей-

ствия отображение значений математических констант в MathCad будет следующим:

$$\pi = 3.14159265358979 \quad e = 2.71828182845905 \quad \infty = 1 \times 10^{307} \quad \% = 0.01$$

В процессе вычислений в MathCad значение математических констант может быть переопределено, поэтому необходимо соблюдать стандартную символику в расчетах и следить за ее использованием. Например, если константам π и e присвоить другие значения, то в дальнейших вычислениях они будут иметь последнее присвоенное им значение:

$$\pi:=23 \qquad e:=46 \qquad \frac{\pi}{10} = 2.3 \qquad \frac{e}{\pi} = 2$$

Системные переменные

Системные переменные устанавливают точность расчетов при использовании численных методов в операциях интегрирования, нахождения корней уравнений или решении систем уравнений, задают значение нижнего индекса массива, устанавливают параметры ввода - вывода данных. Всего в MathCad шесть системных переменных, которые представлены в таблице 2.3.

Таблица 2.3 Системные переменные MathCad

Наименование	*Обозначение*	*Описание*	*Значение по умолчанию*
Начальный индекс массива	*ORIGIN*	Определяет начальный индекс массива.	*ORIGIN=0*
Точность сходимости алгоритма	*TOL*	Допустимая погрешность для различных алгоритмов интегрирования, решения уравнений и т. д.	*TOL=0.001*
Граничная точность	*CTOL*	Критерий точности для задач, в которых используется блок Given	*CTOL=0.001*
Начальная величина для случайных чисел	*Seed value of random numbers*	Параметр, определяющий работу генератора случайных чисел	*Начальное значение=1*
Количество значащих цифр, учитываемых при записи файла	*Pmprecision*	Количество значащих цифр, записываемых в файл функцией writeprn	*Pmprecision=4*
Формат столбца при выводе в файл	*Pmcolwidth*	Ширина столбца (в символах), учитываемая при записи в файл функцией writeprn	*Pmcolwidth=8*

Изменение значений системных переменных осуществляется посредством меню ***Математика, Опции.*** В диалоговом окне Math Options можно просмотреть или отредактировать значения системных переменных (рис.2.4). Влияние значения системных переменных на результаты работы некоторых функций приведено на листинге 2.2. Для функции ***rnorm(n,μ,)*** , обеспечивающей генерацию вектора n случайных чисел, подчиняющихся нормальному закону

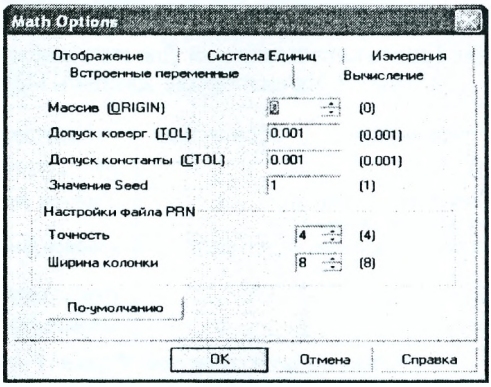


Рис. 2.4. Значения системных переменных

распределения с математическим ожиданием =μ и стандартным отклонением =σ, при различных значениях *Seed* генерируются различные наборы данных.

Листинг 2.2. **Влияние системных переменных на результаты вычислений**

При параметрах seek=1 n=10 m=0 s=1

morm (10, 0, 1) =

	0
0	-1.21
1	-1.757
2	-0.218
3	0.563
4	1.368
5	0.813
6	-0.56
7	0.638
8	-0.241
9	1.622

При параметрах seek=10 n=10 m=0 s=1

morm (10, 0, 1) =

	0
0	-0.311
1	0.02
2	-0.774
3	0.03
4	-1.654
5	0.64
6	0.127
7	0.794
8	-0.172
9	0.682

Массивы

Данные, представленные в виде столбца, получили в MathCad наименование вектор, а данные, представленные в виде таблицы, получили название матрица. Общее название для вектора и матрицы, аналогично с классической математикой, - массив.

Массив можно задать несколькими способами:

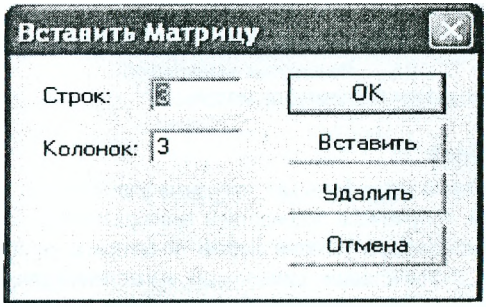


Рис. 2.5. Задание размеров массива

- ввести команду ***Вставка, Матрица***;
- нажать комбинацию клавиш ***Ctrl + M***;
- щелкнуть мышью по кнопке "Вектор или матрица" на панели инструментов *"Матрица"*. Для задания размеров массива в диалоговом окне *"Вставить Матрицу"* (рис.2.5) указать требуемое количество колонок (столбцов) и строк. Допускается редактирование массива путем добавления или удаления строк и столбцов. Для выполнеия операции редактирования массива необходимо поместить курсор рядом с элементом массива, правее которого вставляется (удаляется) колонка или ниже которого вставляется (удаляется) строка. Активизируется диалоговое окно *"Вставить Матрицу"*, задается количество колонок или строк и выполняется команда *"Вставить"* или *"Удалить"*.

При расчетах в Mathcad обычно оперируют именами массивов и дальнейшие операции выполняются над именами массивов. Для обращения к элементу массива используются нижние индексы. По умолчанию индексация элементов массива начинается с 0. Чтобы установить начальную индексацию элементов масссива с 1, необходимо

воспользоваться диалоговым окном *"Math Options"* или изменить значение параметра ***ORIGIN*** в тексте задачи. Например:

$$\mathbf{ORIGIN} := 1 \qquad \mathbf{A} := \begin{pmatrix} 5 & 6 & 7 \\ 1 & 2 & 3 \\ 8 & 9 & 4 \end{pmatrix} \qquad \mathbf{B} := \begin{pmatrix} -1 & 4 & -5 \\ 2 & 4 & 9 \\ 1 & -3 & 8 \end{pmatrix} \qquad \mathbf{C} := \mathbf{A} + \mathbf{B}$$

$$\mathbf{C} = \begin{pmatrix} 4 & 10 & 2 \\ 3 & 6 & 12 \\ 9 & 6 & 12 \end{pmatrix} \qquad \mathbf{C}_{1,1} = 4 \qquad \mathbf{C}_{3,3} = 12 \qquad \frac{\mathbf{C}_{2,3}}{\mathbf{C}_{1,3}} = 6$$

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Когда используется символ присваивания и знак равенства?
2. Что такое встроенные функции и функции пользователя?
3. Как перевести значение аргумента тригонометрической функции в радианы, если оно задано в градусах?
4. Как при выводе могут отображаться числа, и где можно указать формат отображения результатов расчетов?
5. Какие встроенные функции используются для работы с комплексными числами?
6. Перечислите математические константы MathCad, и как они обозначаются?
7. Какие переменные MathCad относятся к встроенным системным переменным и для чего они используются?
8. Как задать массив?
9. С помощью какого параметра можно установить или изменить нижнюю индексацию элементов массива?
10. Как ввести нижний индекс при обозначении элементов массивов?

2.3. СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ

Результатом математических вычислений могут быть как числовые значения, так и символьные (аналитические) выражения, полученные посредством преобразования исходных математических выражений, описывающих условия задачи и алгоритм ее решения в символьном виде. Основными достоинствами символьных вычислений являются отсутствие погрешности вычислений и универсальность использования в различных прикладных задачах.

2.3.1. ВЫПОЛНЕНИЕ СИМВОЛЬНЫХ ВЫЧИСЛЕНИЙ

В Mathcad реализовано три основных способа выполнения символьных вычислений:

- при помощи меню *"Символика"* (рис. 2.6), обеспечивающего возможность выполнения самых разнообразных символьных вычислений (решение и преобразование символьных выражений, интегрирование, дифференцирование и т.д.);
- при помощи специального оператора символьного ввода, включающего знак символьного равенства [→], иногда с добавлением операторов панели инструментов *"Символика"* (рис.2.1);

- с использованием стандартных функций, в сочетании с символьным знаком равенства.

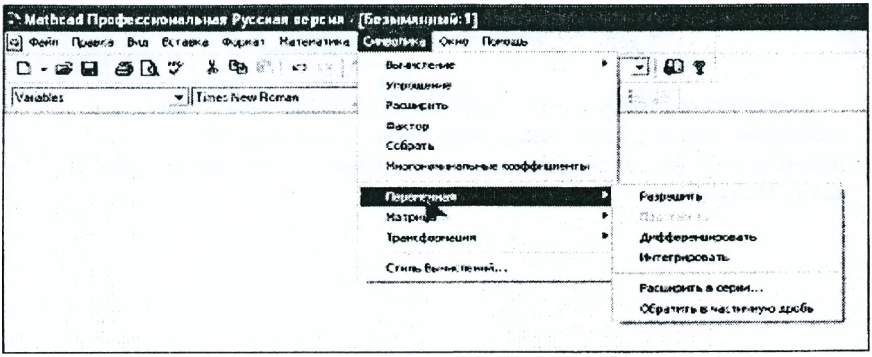


Рис. 2.6. Команды меню "Символика"

2.3.2. СИМВОЛЬНАЯ АЛГЕБРА

Упрощение и разложение выражений, приведение подобных, разложение на множители

Операции символьной алгебры можно рассмотреть в сочетании с командами рабочей панели *"Символика"*, реализующими данную операцию. Перечень некоторых команд и краткое их назначение приведены в таблице 2.4.

Решение уравнений и систем уравнений

Для решения уравнений в символьном виде в MathCad используется специальный оператор символьного решения ***solve*** в сочетании со знаком символьного равенства, который может быть также введен с рабочей панели *"Символика"*. Например,

Таблица 2.4 Основные команды символьного меню, реализующие упрощение и приведение выражений

Обозначение	*Описание*	*Пример*
Simplify	Упрощение алгебраических выражений	$\frac{(x^3 + y^3)}{x + y}$ simplify $\rightarrow x^2 - y \cdot x + y^2$
Collect	Приведение подобных членов	$a \cdot x^2 + b \cdot x^2 - c x$ collect, $x \rightarrow (a + b) \cdot x^2 - c$
Factor	Разложение на множители	$x^3 - y^3$ factor $\rightarrow (x - y) \cdot (x^2 + y \cdot x + y^2)$
Expand	Разложение выражений в более простые суммы	$(x - 1)^4$ expand, $x \rightarrow x^4 - 4 \cdot x^3 + 6 \cdot x^2 - 4 \cdot x + 1$
Parfrac	Разложение на элементарные дроби	$\frac{(x^3 - 2 \cdot x^2 + 1)}{(x^2 + 1)}$ convert, parfrac, $x \rightarrow x - 2 + \frac{(3 - x)}{(x^2 + 1)}$
Substitute	Подстановка переменной	$2 \cdot x^2 - x + 4$ substitute $, x = \frac{1}{y^2} \rightarrow \frac{2}{y^4} - \frac{1}{y^2} + 4$
Coeffs	Формирование вектора коэффициентов полинома для последующего определения корней полинома.	$P := 2 \cdot x^3 - 3 \cdot x^2 - 4 \cdot x + 2$ P coeffs $, x \rightarrow \begin{pmatrix} 2 \\ -4 \\ -3 \\ 2 \end{pmatrix}$

$$2x^2 + 3 \cdot x - 4 \text{ solve}, x \rightarrow \begin{pmatrix} \frac{-3}{4} + \frac{1}{4} \cdot \sqrt{41} \\ \frac{-3}{4} - \frac{1}{4} \cdot \sqrt{41} \end{pmatrix}$$

Аналогичные действия можно выполнить, используя меню "Символика". Для этого необходимо записать вычисляемое выражение. Затем выделить переменную, относительно которой решается уравнение, войти в меню ***Символика, Переменная, Разрешить***. Например,

$$2 \cdot x^2 + 3 \cdot x - 4$$

$$\begin{pmatrix} \frac{-3}{4} + \frac{1}{4} \cdot \sqrt{41} \\ \frac{-3}{4} - \frac{1}{4} \cdot \sqrt{41} \end{pmatrix}$$

В случае, если необходимо упростить полученный результат, используется знак равенства **[=]**. Например,

$$3 \cdot x^2 + 14 \cdot x - 12 \text{ solve}, x \rightarrow \begin{pmatrix} \frac{-7}{3} + \frac{1}{3} \cdot \sqrt{85} \\ \frac{-7}{3} - \frac{1}{3} \cdot \sqrt{85} \end{pmatrix} = \begin{pmatrix} 0.74 \\ -5.407 \end{pmatrix}$$

При решении некоторых уравнений, результат включает большое количество символов. Mathcad сохраняет его в буфере, а на дисплей выводится сообщение: "*This array has more elements than can be displayed at one time. Try using the "submatrix" function" – "Этот массив содержит больше элементов, чем может быть отображено одновременно. Попытайтесь использовать функцию "submatrix"*". В этом случае рекомендуется использовать численное решение. Или, в случае необходимости, символьное решение может быть выведено и отображено на дисплее.

Символьное решение может быть получено с использованием блока ***given ... find***. В этом случае при записи уравнения для связи его левой и правой части использует символ логического равенства "=" с панели инструментов Boolean, например,

Given

$$3 \cdot x^2 + 14 \cdot x - 12 = 0$$

$$\text{Find}(x) \rightarrow \begin{pmatrix} \frac{-7}{3} + \frac{1}{3} \cdot \sqrt{85} & \frac{-7}{3} - \frac{1}{3} \cdot \sqrt{85} \end{pmatrix}$$

Аналогичным способом решаются системы уравнений в символьном виде. Ниже приводятся примеры решения систем уравнений в символьном виде различными способами. При использовании оператора символьного решения solve в сочетании со знаком символьного равенства ■ **solve** , ■ → система уравнений должна быть задана в виде вектора, который вводится вместо левого маркера оператора solve, а перечень переменных, относительно которых решается система, вместо правого маркера. Например,

$$\begin{pmatrix} x^2 + y = 41 \\ x + y = 11 \end{pmatrix} \text{solve},x,y \rightarrow \begin{pmatrix} 6 & 5 \\ -5 & 16 \end{pmatrix}$$

Пример использования блока ***given...find*** для решения системы уравнений:.

Given

$$x^2 - y^2 = x - 3$$

$$2 \cdot x + y = 4$$

$$\text{Find}(x,y) \rightarrow \begin{pmatrix} \frac{5}{2} + \frac{1}{6} \cdot \sqrt{69} & \frac{5}{2} - \frac{1}{6} \cdot \sqrt{69} \\ -1 - \frac{1}{3} \cdot \sqrt{69} & -1 + \frac{1}{3} \cdot \sqrt{69} \end{pmatrix}$$

2.3.3. МАТЕМАТИЧЕСКИЙ АНАЛИЗ

Дифференцирование и интегрирование

Символьное дифференцирование в MathCad осуществляется или посредством выполнения самой операции над задаваемым выражением, или с использованием и отображением оператора производной. В первом случае достаточно ввести выражение, выделить переменную дифференцирования в выражении, войти в меню ***Символика, Переменная, Дифференцировать***. Несмотря на оперативность выполнения данной операции, при оформлении отчета о данных вычислениях, возникают неудобства, связанные с отсутствием самого символа (оператора) производной. Например,

$$\sin(x)^2 - 2 \ln(x)$$

Результат дифференцирования посредством меню ***Символика***

$$2 \cdot \sin(x) \cdot \cos(x) - \frac{2}{x}$$

Аналогичного результата можно добиться, используя символику дифференцирования на панели инструментов *"Исчисление"*. Данный вариант является более предпочтительным при формировании отчета о результатах вычислений:

$$\frac{d}{dx}\left(\sin(x)^2 - 2 \cdot \ln(x)\right) \rightarrow 2 \cdot \sin(x) \cdot \cos(x) - \frac{2}{x}$$

MathCad позволяет вычислить производную n-го порядка, что также можно осуществить используя панель инструментов "Исчисление":

$$\frac{d^3}{dx^3}\left(\pi \cdot x^4 + \frac{v}{x}\right) \rightarrow 24 \cdot \pi \cdot x - 6 \cdot \frac{v}{x^4}$$

Аналогично символьному дифференцированию выполняется и символьное интегрирование. Причем можно вычислить как определенный, так и неопределенный интеграл, используя команды меню *"Символика"* (***Символика, Переменная, Интегрировать***) или элементы панели инструментов "Символика". При этом необходимо учитывать, что в результате вычисления неопределенного интеграла постоянная интегрирования автоматически не выводится, а должна вводится пользователем принудительно. На Листинге 2.3. приводится пример символьного интегрирования.

Необходимо отметить, что при вычислениях с помощью операторов панели инструментов "Символика" и команд меню "Символика" могут получаться разные результаты. Это связано с тем, что команды меню применяются только к выделенному выражению, в то время как при использовании операторов панели инструментов "Символика" учитываются все предшествующие вычисления.

Ряды. Сумма и произведение ряда

Сумма членов ряда обозначается в MathCad так же как и в классической математике, символом $\sum$. Аналогично произведение членов ряда обозначается символом $\prod$. Вместе с тем следует различать две разновидности вычисления сумм и произведений членов ряда, предусмотренных в MathCad.

В первом случае обозначения содержат по четыре маркера, назначение которых со-

Листинг 2.3. **Примеры символьного интегрирования**

Вычисление неопределенного ентеграла

1). С использованием меню "Символика"

$$\sqrt{\sin(x)}\cdot\cos(x)+\ln(x)$$

перед выполнением операции интегрирования предварительно выделяется переменная интегрирования x

$$\frac{2}{3}\cdot\frac{\sin(x)^3}{2}+x\cdot\ln(x)-x+\tilde{N}1$$

К полученному результату принудительно приписывается постоянная интегрирования C1

2) С использованием элементов панели инструментов "Исчисление"

$$\int\sqrt{\sin(x)}\cdot\cos(x)+\ln(x)\,dx\rightarrow\frac{2}{3}\cdot\sin(x)^{\frac{3}{2}}+x\cdot\ln(x)-x$$

$$\frac{2}{3}\cdot\frac{\sin(x)^3}{2}+x\cdot\ln(x)-x+\tilde{N}1$$

Для проведения дальнейших вычислений к результату так же приписывается C1

Вычисление определенного интеграла

$$\int_{\frac{\pi}{6}}^{\frac{\pi}{2}}\left(\sqrt{\sin(x)}\cos(x)+x\right)dx\rightarrow\frac{2}{3}+\frac{1}{9}\cdot\pi^2-\frac{1}{6}\cdot\sqrt{2}=1.528$$

ответствуют принятым в математике правилам. Пример записи суммы (произведения) членов ряда приведен на листинге 2.4.

Во втором случае обозначения содержат всего лишь два маркера, а сама операция предназначена для так называемых ранжированных переменных. Имя ранжированной переменной (k) указывается в нижнем индексе, а ее диапазон задается дополнительно.

Если переменная не определена, то она выбирается из условия ее изменения, от 1 до текущего значения переменной k.

Листинг 2.4. **Примеры вычисления суммы и произведения ряда**

Вычисление суммы ряда

$$\sum_{\blacksquare=\blacksquare}^{\blacksquare}\blacksquare \qquad \sum_{k=1}^{\infty}\frac{1}{(k+1)\cdot(k+2)}\rightarrow\frac{1}{2} \qquad \sum_{k=0}^{\infty}\frac{x^k}{k!}\rightarrow\exp(x)$$

Вычисление суммы ряда для ранжированных переменных

$n := 1..5$

$$\sum_{\blacksquare}\blacksquare \qquad \sum_{n}\left[\frac{n}{(10\cdot n+1)}\right]\rightarrow\frac{796135}{1663739}=0.479 \qquad \sum_{k}\frac{1}{k\cdot(k+1)}\rightarrow\frac{-1}{k}$$

Вычисление произведения ряда

$$\prod_{i=1}^{5}\frac{\pi i}{i+1}\rightarrow\frac{1}{6}\cdot\pi^5$$

Вычисление произведения для ранжированного ряда

$$\prod_{k}\frac{1}{k}\rightarrow\frac{1}{\Gamma(k)}$$

где Γ(k) - гамма - функция Эйлера

$$\tilde{A}(k):=\int_0^{\infty}t^{k-1}\cdot e^{-t}\,dt$$

Вычисление пределов

Введем несколько понятий, которые будут необходимы нам в дальнейшем. Число *m* является пределом числовой последовательности *{a_i}*, если для любого положительного числа ε найдется зависящее от него натуральное число, *N* такое, что при всех $n > N$ выполняется неравенство $|a_n - m| < \varepsilon$. При этом пишут:

$$\lim_{n\rightarrow\infty} a_n = n$$

Число *m* является пределом функции *f(x)* в конечной или бесконечной точке x_0, если для любой последовательности x_n, $n = 1,2,\ldots$, стремящейся к x_0, последовательность $f(x_n)$, $n = 1,2,\ldots$стремится к точке *m*. В этом случае пишут

$\lim_{x\rightarrow x_0} f(x) = m$ Если $\lim_{x\rightarrow x_0^+} f(x) = f(x_0)$, то говорят о пределе функции y=f(x) справа. Если $\lim_{x\rightarrow x_0^-} f(x) = f(x_0)$, то говорят о пределе функции слева.

Аналогичная символика принята в MathCad. При этом для вычисления предела последовательности и функции, предела функции слева и справа приняты следующие обозначения:

$$\lim_{\blacksquare\rightarrow\blacksquare}\blacksquare \qquad \lim_{\blacksquare\rightarrow\blacksquare^+}\blacksquare \qquad \lim_{\blacksquare\rightarrow\blacksquare^-}\blacksquare$$

При вычислении пределов используется только знак символьного равенства. В процессе проведения вычислений, кроме результатов, могут выдаваться различные сообщения. Если ошибочно введено выражение, для которого предстоит вычислить пре-

дел, то MathCad выводит сообщение: *" No symbolic result was found → Символический результат не был найден"*. Если MathCad не может вычислить выражение, то в качестве ответа повторяется само условие задачи. Если предел не существует, то результатом вычислений будет выражение: *"Undefined – Не определен"*. Пример вычисления пределов приведен на листинге 2.5.

Листинг 2.5. **Примеры вычисления пределов**

1). Предел числовой последовательности

$$\lim_{n \to \infty} \prod_{k=1}^{n} \left(1 + \frac{k}{n^2}\right) \to \exp\left(\frac{1}{2}\right)$$

2) Предел функции

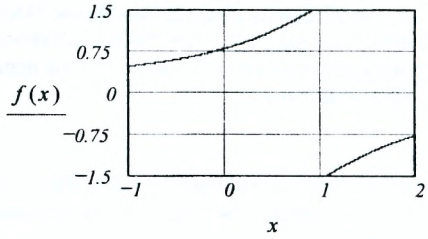
$$\lim_{x \to \infty} \frac{\ln\left(2 + e^{3 \cdot x}\right)}{\ln\left(3 + e^{2 \cdot x}\right)} \to \frac{3}{2}$$

3) Предел функции слева и справа

В качестве примера рассмотрим функцию arctg(1/(1-x))

График функции имеет вид

$$f(x) := \operatorname{atan}\left(\frac{1}{1 - x}\right)$$



$$\lim_{x \to 1^{+}} \frac{1}{1 - x} \to -\infty$$

$$\lim_{x \to 1^{-}} \frac{1}{1 - x} \to \infty$$

Разложение в ряд Тейлора

Для разложения функции f(x) в ряд Тейлора используется оператор *series* панели инструментов *"Символика"*. Оператор имеет вид , series, , , . Самое левое поле предназначено для ввода функции. Среднее поле должно содержать имя переменной, по которой производится разложение функции в ряд Тейлора, или выражение вида " *x = a*", где *x* – идентификатор переменной, *a* – точка, в которой производится разложение .Если значение *a* не задается, то по умолчанию оно принимается равным 0. В крайнее правое поле вводится число, определяющее порядок многочлена Тейлора. Примеры разложения функций в ряд Тейлора приведены на листинге 2.6.

Листинг 2.6. **Разложения функций в ряд Тейлора**

$e^x \text{ series } ,x ,4 \rightarrow 1 + x + \frac{1}{2} \cdot x^2 + \frac{1}{6} \cdot x^3$

$e^x \text{ series } ,x = 4 ,3 \rightarrow \exp(4) + \exp(4) \cdot (x - 4) + \frac{1}{2} \cdot \exp(4) \cdot (x -$

$(\ln(x + 1)) \text{ series } ,x = 2 ,3 \rightarrow \ln(3) + \frac{1}{3} \cdot x - \frac{2}{3} - \frac{1}{18} \cdot (x - 2)^2$

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какими способами можно выполнять символьные вычисления в MathCad?
2. Как отображается знак символьного равенства?
3. Перечислите операции символьной алгебры, и как они представлены на рабочей панели "Символика"?
4. Какими способами можно получить символьное решение уравнения и системы уравнений?
5. Как используется блок *given ... find* для получения символьного решения уравнения и системы уравнений?
6. Укажите, как можно осуществить операцию символьного дифференцирования?
7. В чем отличие между двумя способами вычисления сумм и произведений членов ряда в MathCad?
8. Какие три разновидности вычисления пределов предусмотрены в MathCad?
9. Какой оператор используется для разложения функции в ряд Тейлора?

2.4. ПОСТРОЕНИЕ ГРАФИКОВ

Для построения графиков в MathCad используются шаблоны. Для начального построения графика достаточно войти в меню ***Вставка, График*** или задать тип графика на панели инструментов "Графики" и выбрать один из семи основных типов графиков:

 декартов график [@] — график в декартовой системе координат ;

 полярный график [Ctrl+ 7] — график в полярной системе координат;

 график поверхности [Ctrl+ 2] — построение трехмерного графика;

 карта линий уровня [Ctrl+ 5] — контурный график поверхностей уровня;

 3D точечный график – график в виде точек в трехмерном пространстве;

 3D столбиковая гистограмма –совокупность столбиков в трехмерном пространстве;

 Векторное поле— график векторного поля на плоскости.

2.4.1. ПОСТРОЕНИЕ ГРАФИКОВ В ДЕКАРТОВОЙ СИСТЕМЕ КООРДИНАТ

Незаполненный шаблон графика представляет собою большой пустой прямоугольник с темными маленькими прямоугольниками, расположенными около осей абсцисс и

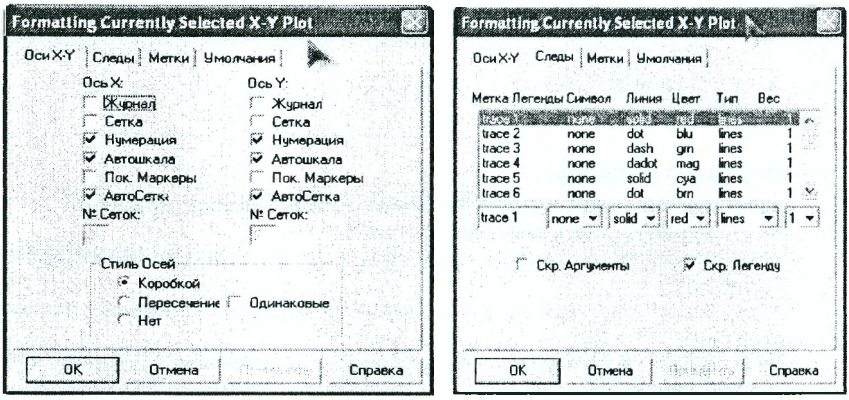
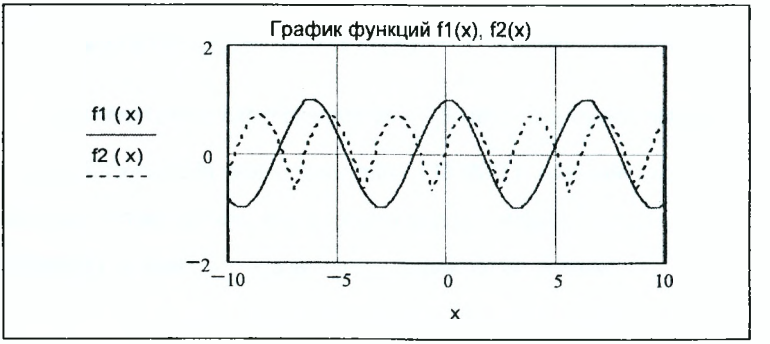


Рис. 2.7. Элементы форматирования графика в декартовой системе координат

координат будущего графика. В средние прямоугольники надо поместить имя аргумента x оси абсцисс и имя функции y оси ординат. Если строятся графики нескольких функций в одном шаблоне, то для их разделения следует использовать запятые. Крайние темные маленькие прямоугольники служат для указания предельных значений абсцисс и ординат, т. е. они задают масштабы графика. Если оставить эти шаблоны незаполненными, то масштабы по осям графика будут устанавливаться автоматически. Но автоматические масштабы могут оказаться не вполне удобными. Поэтому рекомендуется вначале использовать автоматическое масштабирование, а затем изменять их на более подходящие. На листинге 2.7 показаны пример построения графиков.

Чтобы произошло построение графика в автоматическом режиме вычислений, дос-

Листинг 2.7. **Построение графика функции в декартовой системе координат**



таточно вывести курсор за пределы графического объекта. Параметры изображения (цвет и толщина линий, координатная сетка, разметка осей, надписи на графиках и др.) проще всего изменить, щелкнув дважды по полю графика. В результате активизируется диалоговое окно *"Форматирование"*, на котором выбирается соответствующую вкладка и устанавливаются параметры настройки графика (рис. 2.7). Возможно отображение на одном шаблоне графиков функций от различных переменных. В этом случае, количество переменных и функций, их имена и порядок следования должны быть синхронизированы (листинг 2.8).

Листинг 2.8. **Построение графиков функций, зависящих от различных аргументов**

Построение графиков функций в декартовой системе координат

$x := -10, -9.5 .. 10$

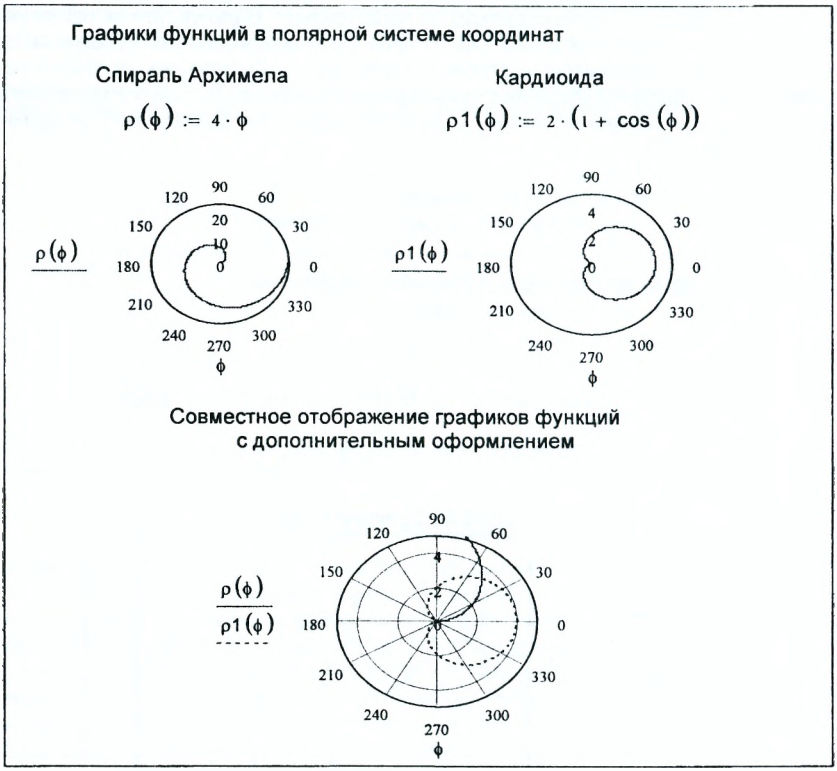
$f1(x) := 3\cos(x)$ $f2(h) := 2\cdot\sin(h) + 3\cos($

$f3(t) := (-t)^3 + 3\cdot t^2 + t - 1$

2.4.2. ПОСТРОЕНИЕ ГРАФИКОВ В ПОЛЯРНОЙ СИСТЕМЕ КООРДИНАТ

Построение графиков в полярной системе координат аналогично построению графиков в декартовой системе координат. Но при этом необходимо учитывать специфику самих функций. В полярной системе координат при активизации шаблона графика, рабочее поле представлено окружностью. В нижней части шаблона задается имя угловой переменной, в левой части - имя функции, определяющей радиус как функцию угла. В правой верхней части расположены два поля для задания нижнего и верхнего значения радиуса. Возможно отображение нескольких функций в рабочем поле графика. Для этого имена функций так же вводятся через запятую. На листинге 2.9 представлены примеры отображения функций в полярной системе координат. Предусмотрена возможность форматирования графиков функций путем вывода шкал радиальных, круговых, вспомогательных линий и т.д. Форматирование обеспечивается с помощью инструментов диалогового окна ***Форматирование***, которое активизируется двойным щелчком мыши по полю графика.

Листинг 2.9. **Отображение графиков в полярной системе координат**



2.4.3. ПОСТРОЕНИЕ ПОВЕРХНОСТЕЙ

Для построения трехмерной поверхности F(x,y), функция предварительно представляется матрицей M ординат F(x,y). При этом выводится шаблон графика, левый верхний угол которого помещается в место расположения курсора. Шаблон содержит единственное поле — темный прямоугольник у левого нижнего угла основного шаблона. В него надо занести имя матрицы M или имя функции F при автоматическом построении матрицы. Наглядность представления трехмерных поверхностей зависит от множества факторов: масштаба построений, углов поворота фигуры относительно осей, применения алгоритма удаления невидимых линий или отказа от него, использования функциональной закраски и т.д. Для изменения этих параметров следует использовать операцию установки формата графика. При построении трехмерных поверхностей и объемных фигур можно использовать параметрическое задание описывающих их функций. Фигуры задаются значениями координат *x*, *y* и *z* всех точек фигуры. При этом в шаблоне 3D-графики указываются три матрицы, хранящие массивы этих координат, — X, Y и Z. На листинге 2.10 приведены примеры построения графиков поверхностей.

Листинг 2.10. **Построение поверхностей, задаваемых функцией от двух переменных**

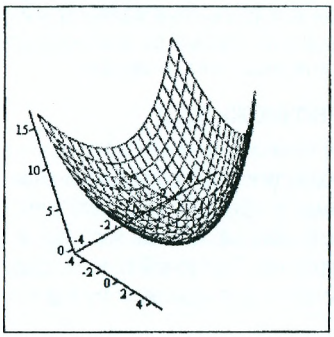
Построение поверхностей

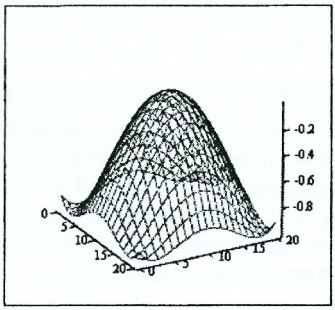
$$F(x,y) := \frac{x^2 + y^2}{3}$$

$$F1(x,y) := -\sin\left(x^2 + y^2\right)$$

$$x := 0..20 \qquad y := 0..20$$

$$M_{x,y} := F1\left(\frac{x-10}{10}, \frac{y-10}{10}\right)$$





F M

Можно изменять заданные по умолчанию параметры графиков. Для этого необходимо вызвать окно диалога форматирования трехмерных графиков (3-D) (рис. 2.8, 2.9) двойным щелчком мыши по полю графика. Диалоговое окно *3-D Plot Format* содержит множество флажков для вы-бора режима построения гра-фика и девять закладок: *Backplanes (Основание); Special (Специальный); Advanced (Допол-нительно); QuickPlotData (Графи-ческие данные); General (Общее); Axes (Ось); Appearance (Внешний Вид); Lighting (Освещение); Title (Название).*

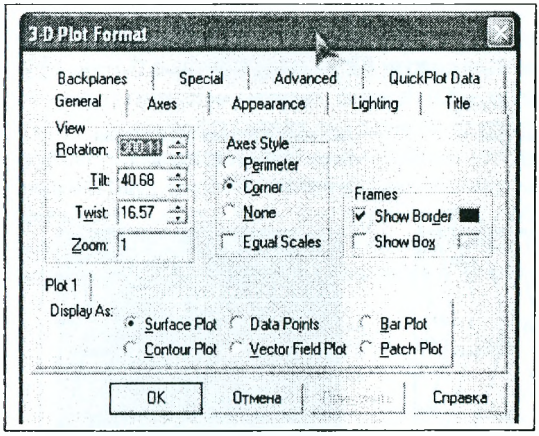


Рис. 2.8. Диалоговое окно "Форматирование трехмерных графиков"

Ограничимся рассмотрением одной вкладки, представленной на рис. 2.8, - *General (Общее).* Первый комплект чисел в раз-деле *View (Вид)* показывает *Rolation (Вра-*

щение), Tilt (Нак-лон), Twist (Искривление), Zoom (Масштаб), под которыми наблюдается построенный график поверхности. Далее в разделе *Axes Style (Стиль оси)* имеется ряд переключателей и флажок для выбора стиля изображения размеров графика:

- *Perimetr (периметр)* – выводит график с размерами по периметру;
- *Corner (угол)* – выводит график с размерами по осям;
- *None (нет)* – выводит график без размеров по периметру и по осям;
- *Equal scales (равные шкалы)* – установка по осям равных масштабов.

В пункте *Frames (Границы графика)* определяется обрамление графика:

- *Border (границы)* – показывает границы графика;
- *Show Box (каркас)* – показывает график в виде параллелепипеда.

На панели переключателей *Plot 1 (График 1)* можно выбрать одну из форм представления трехмерного графика. При работе с панелями настройки параметров изображения можно посмотреть результат, полученный при изменении параметра изображения, не закрывая панели. Для этого после изменения параметра щелкните по кнопке *Применить*. Для возвращения в документ щелкните мышью по кнопке **OK**.

2.4.4. ДОПОЛНИТЕЛЬНЫЕ ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ

В данном разделе рассмотрены графические возможности MathCad по созданию контурного графика - поверхностей одинакового уровня (изолинии); графика в виде точек в трехмерном пространстве; столбиковой гистограммы и графика векторного поля на плоскости. Данные графические зависимости носят специализированный характер, и это предопределяет их более узкое использование на практике. Контурный график представляет собой совокупность линий, каждая из которых соответствует одинаковому значению функ-

Листинг 2.11. **Построения плана изолиний**

Построение плана изолиний

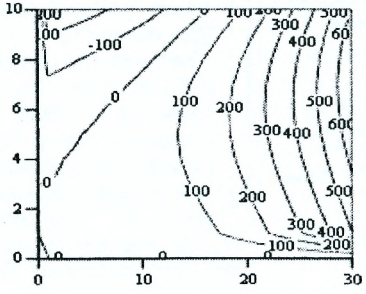
$$f(x,y) := \left(\frac{2}{3}x^3 - 3y^3 + 10x\cdot y\right) - 4x + 3y^2$$

$i := 1 .. 30$ $\quad j := 1 .. 10$

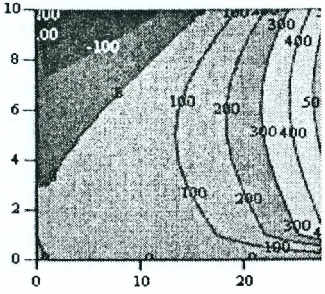
$x_i := 0.3\cdot i$ $\quad y_j := 0.5\cdot j$

$M_{i,j} := f(x_i, y_j)$

Активизированы параметры Draw Lines, Auto Contour, Numbered



Активизированы параметры Draw Lines, Auto Contour, Numbered, Fill



ции, зависящей от двух переменных (изолинии). Такие функции получили широкое распространение в картографии, геодезии, океанологии, экологии и т. д. Последовательность построения контурного графика следующая. Сначала вводится функция двух переменных f(x,y). Далее определяются значения x_i, y_j, задающие дискретные точки по осям x, y. Заполняется матрица M значениями $f(x_i,y_j)$. Отображается матрица M в виде карты изолиний. На листинге 2.11 приведен пример построения контурного графика.

Форматирование изображения (количество линий уровня их значения, заливка) производится посредством диалогового окна *3-D Plot Format (Форматирование)*, которое представлено на рис. 2.9. Переключатели диалогового окна позволяют сформировать дизайн графика. Например, переключатели группы *Contour Options (Контурные опции)* устанавливают следующие опции графика: *Fill (Залить)* – обеспечивают закраску графика согласно цветовой палитры. *Draw Lines (Рисовать линии)* – позволяет отображать на графике лини уровня. *Auto Contour (Автоконтур)* – количество линий контура выбирается автоматически. *Numbered (Пронумерованные)* – на линии уровня выносятся их числовые значения.

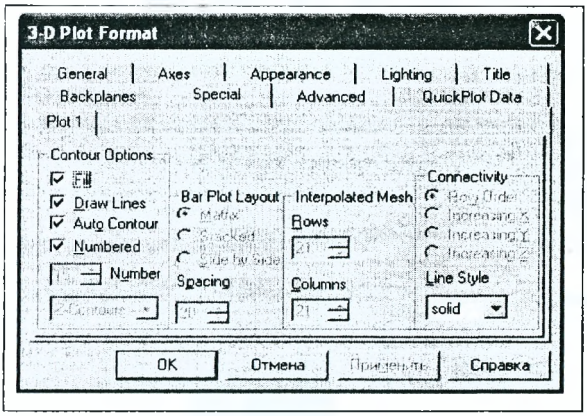


Рис. 2.9. Диалоговое окно форматирования графика (вкладка - Специальная)

На листинге 2.12. представлены графики функций, демонстрирующие графические возможности MathCad при построении столбиковых гистограмм, точечных диаграмм, векторного поля. Один вид 3-D диаграмм функции 2-х переменных может быть трансформирован в другой вид посредством вкладки *General (Общее)* диалогового окна *3-D Plot Format* (рис.2.9). Для преобразования диаграммы в другой вид необходимо выделить график и установить соответствующий переключатель группы *Display As*. При этом диаграмма принимает вид, соответствующий установленному переключателю: *Surfase Plot* – график поверхности. *Contour Plot* – контурный график. *Data Points* – точечный график. *Vector Fields Plot* – векторное поле. *Bar Plot* – столбиковая гистограмма. *Patch Plot* – "Кусочный" график (совмещенный график поверхности и точечный график).

Листинг 2.12. **Примеры 3-D диаграмм**

Функция от 2-х переменных, определяющая значения элементов исходной матрицы

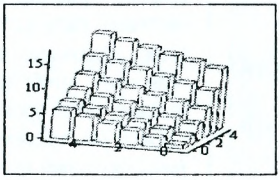
$i := 0 .. 5 \qquad j := 0 .. 5 \qquad x_i := (i + 1) \qquad y_j := 2 \cdot j + 1$

$f(x, y) := \frac{x^2}{3} + \frac{y}{2}$

$M_{i,j} := f(x_i, y_j)$

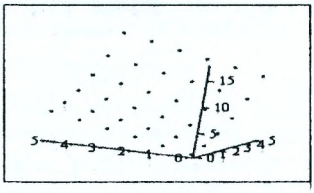
$$M = \begin{pmatrix} 0.833 & 1.833 & 2.833 & 3.833 & 4.833 & 5.833 \\ 1.833 & 2.833 & 3.833 & 4.833 & 5.833 & 6.833 \\ 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 \\ 5.833 & 6.833 & 7.833 & 8.833 & 9.833 & 10.833 \\ 8.833 & 9.833 & 10.833 & 11.833 & 12.833 & 13.833 \\ 12.5 & 13.5 & 14.5 & 15.5 & 16.5 & 17.5 \end{pmatrix}$$

3-D столбиковая гистограмма



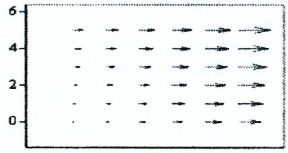
M

3-D точечная диаграмма



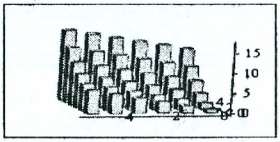
M

Векторное поле



M

3-D столбиковая гистограмма с уменьшенными размерами элемента



M

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие типы графиков можно строить в MathCad?
2. Как отобразить на одном графике в декартовой системе координат несколько функциональных зависимостей от одной переменной и от разных переменных?
3. Какие поля для задания параметров графика предусмотрены в полярной системе координат?
4. Какие закладки предусмотрены в диалоговом окне 3-D Plot Format?
5. Как задать количество линий уровня для контурного графика?
6. Как преобразовать 3 –D диаграмму в другой вид?

2.5. ЧИСЛЕННЫЕ ВЫЧИСЛЕНИЯ

2.5.1. ТАБУЛИРОВАНИЕ ФУНКЦИЙ

Под табулированием функций понимается вычисление дискретных значений функции при изменении значения аргумента по закону арифметической прогрессии. При этом функция должна быть непрерывной на отрезке табулирования. Результаты табулирования принято представлять в виде таблиц. При табулировании необходимо определить значение дискретного аргумента, для чего задается идентификатор дискретного аргумента и определяется область его значений. Одним из способов задания значений

Листинг 2.13. **Табулирования функций**

Табулирование функций

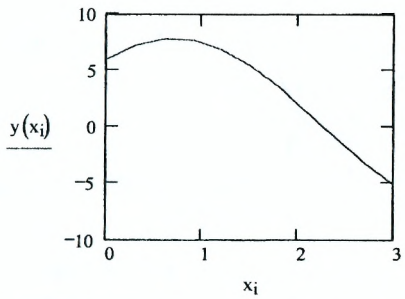
Задание функции 1 $y(x) := 5 \cdot \sin(x) + 6 \cdot \cos(x)$

Задание счетчика значений дискретного аргумета i $i := 0 .. 10$

Задание значений дискретного аргумента $x_i := 3 - 0.3 \cdot i$

График функции y(x) Вычисление значений функции

i =	x_i =	$y(x_i)$ =
0	3	5.234
1	2.7	3.288
2	2.4	1.047
3	2.1	1.287
4	1.8	3.506
5	1.5	5.412
6	1.2	6.834
7	0.9	7.646
8	0.6	7.775
9	0.3	7.21
10	0	6

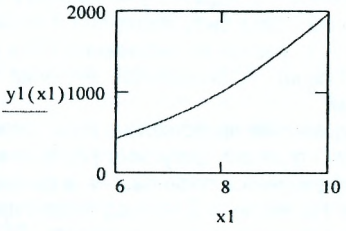


Задание функции 2 $y1(x1) := 2 \cdot x1^3 - 3x1 + 6$

Задание значений дискретного аргумента $x1 := 6, 6.5 .. 10$

x1 =
6
6.5
7
7.5
8
8.5
9
9.5
10

График функции y1(x1)



Вычисление значений функции

y1(x1) =
420
535.75
671
827.25
$1.006 \cdot 10^3$
$1.209 \cdot 10^3$
$1.437 \cdot 10^3$
$1.692 \cdot 10^3$
$1.976 \cdot 10^3$

дискретного аргумента, является задание счетчика значений дискретного аргумента или иначе – ранжированной переменной. Изменение аргумента задается в формате

x:=начальное значение[,начальное значение+шаг]..конечное значение

в скобках указан необязательный параметр, если его нет, шаг, по умолчанию, равен 1. Двоеточие ".." вводится символом точка с запятой ";" или кнопкой "m..n" панели инструментов "Матрицы". В некоторых случаях возможно непосредственное задание значений дискретного аргумента, при этом нет необходимости задавать счетчик значений дискретного аргумента. Для каждого значения дискретного аргумента определяется значение функции. Пример табулирования функций представлен на листинге 2.13.

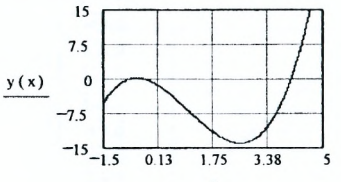
2.5.2. ЧИСЛЕННОЕ РЕШЕНИЕ УРАВНЕНИЙ С ОДНИМ НЕИЗВЕСТНЫМ

Листинг 2.14. **Решение уравнений с использованием функций root( ) и polyroots( )**

Решение уравнений с одним неизвестным с использованием функции root( ) и polyroots( )

Решить уравнение вида $x^3 - 3 \cdot x^2 - 4 \cdot x - 1 = 0$

График функции $y(x) := x^3 - 3 \cdot x^2 - 4 \cdot x - 1$



$$v := \begin{pmatrix} -1 \\ -4 \\ -3 \\ 1 \end{pmatrix} \quad \text{polyroots}(v) = \begin{pmatrix} -0.692 \\ -0.357 \\ 4.049 \end{pmatrix}$$

$x := 5$ root $(y(x), x) = 4.049$

$x := -1$ root $(y(x), x) = -0.692$

$x := 0$ $TOL := 10^{-10}$ root $(y(x), x) = -0.357$

$x := 0$ $TOL := 10^{-1}$ root $(y(x), x) = -0.308$

$x := -0.3$ root $(\sin(x) - x, x) = -0.209$

Простейший способ найти корень уравнения с одним неизвестным обеспечит функция ***root ( ).*** Аргументами функции ***root ( )*** являются вид функции определяющей решаемое уравнение и имя переменой, относительно которой ищется решение - ***root (f(x),x)*** Если уравнение содержит несколько корней, то функция обеспечивает нахождение единственного корня, ближайшего к заданному начальному значению для искомой переменной. Точность вычислений может быть увеличена или уменьшена посредством задания значения переменной *TOL*, равной по умолчанию 10^{-3} и определённой в меню *Math, Options (Математика, Опции)*. Установленное значение *TOL* также оказывает влияние на точность вычислений.

В случае, если решаемое уравнение представлено полиномом, то все его решения могут быть получены с помощью функции ***polyroots (v).*** В качестве аргументов этой функции выступает вектор коэффициентов полинома –v, а результат представляется в виде вектора корней полинома. На листинге 2.14 представлен пример нахождения корней уравнений с использованием функций ***root ( ) и polyroots ( ).***

Листинг 2.15. **Решение уравнений с использованием блоков given...find( ) и given...minerr ( )**

Решение уравнений с одним неизвестным с использованием блока given...find( ) и given...minerr( )

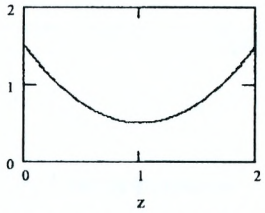
Нахождение корней уравнения cos(x) - sin(x) = 0

x := −1 Given cos (x) − sin(x) = 0 Find (x) = −2.356

x1 := 1 Given cos (x1) − sin(x1) = 0 Find (x1) = 0.785

Нахождение решения уравнения $y(z) := z^2 - 2 \cdot z + 1.5$

с использованием блока given...minerr( )



z := 0

Given

$z^2 - 2 \cdot z + 1.5 = 0$

Minerr (z) = 1

ERR = 0.25

Нахождение решения уравнения с использованием блока given...minerr

z1 := 1 Given cos (z1) − sin(z1) = 0 Minerr (z1) = 0.785 ERR = 3.654×10^{-15}

Другим способом решения уравнений является применение специального вычислительного блока, начинающегося с ключевого слова ***given*** с использованием функций ***find( ) и minerr ( )***.

Блок имеет следующую структуру:

Начальное значение искомой переменной
given
Решаемое уравнение
Выражение с использованием функции find( ) или minerr ( )

Нахождение корней уравнения с использованием блока ***given...find ( )*** в чем – то аналогично использованию функции ***root ( ).*** Задается начальное значение для искомой переменной, после находится решение, ближайшее к заданному начальному условию (листинг 2.15).

Использовании блока ***given...minerr ( )*** имеет существенные особенности. Решение будет найдено в любом случае, даже при его отсутствии. Дело в том, что ищется не решение системы, а минимальная невязка уравнений. На листинге 2.15 рассмотрена функция, заведомо не имеющая действительных корней и при использовании блока given...minerr ( ) найдено решение, значение, которое наиболее приближено к оси x, то есть обеспечивает минимальную невязку. Значение невязки (ошибки) показывает системная переменная ***ERR***.

2.5.3. РЕШЕНИЕ СИСТЕМЫ УРАВНЕНИЙ

Первоначально рассмотрим СЛАУ. Для их решения может использоваться блок given ...find( ) или специальная функция lsolve( ). Применение блока given ...find( ) предопределяет необходимость задания начальных значений искомых переменных. Далее после ключевого слова given описывается СЛАУ и с помощью find( ) находится решение. Следует указать, что в том случае, когда СЛАУ имеет бесконечное множество решений блок given ...find( ) дает конкретный результат, что несомненно следует отнести к недостаткам. В случае отсутствия решения будет выдано сообщение *"Matrix is singular. Cannot compute its inversy – Матрица сингулярная. Нельзя вычислить эту инверсию"*.

Применение функции ***lsolve( )*** позволяет избежать этого недостатка. Функция ***lsolve(M,b)*** имеет два аргумента. M – матрица коэффициентов при неизвестных, b – вектор свободных членов. На листинге 2.16 приведен пример решения СЛАУ.

Листинг 2.16. **Решение систем линейных алгебраических уравнений**

Пример решения СЛАУ

$x := 0 \quad y := 0$

Given

$x + 3y = 5$

$2 \cdot x + y = 12$

$\text{Find}(x, y) = \begin{pmatrix} 6.2 \\ -0.4 \end{pmatrix}$

$M := \begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix} \quad b := \begin{pmatrix} 5 \\ 12 \end{pmatrix}$

$\text{lsolve}(M, b) = \begin{pmatrix} 6.2 \\ -0.4 \end{pmatrix}$

Пример решения СЛАУ в случае бесконечного множества решений

$z := 0 \quad p := 0$

Given

$2 \cdot z + 3 \cdot p = 6$

$4 \cdot z + 6 \cdot p = 12$

$\text{Find}(z, p) = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$

$T := \begin{pmatrix} 2 & 3 \\ 4 & 6 \end{pmatrix} \quad k := \begin{pmatrix} 6 \\ 12 \end{pmatrix}$

$\text{lsolve}(T, k) =$

выдается сообщение "Matrix is singular. Cannot compute its inverse"

Для решения системы нелинейных уравнений используются два блока: ***given...find( )*** и ***given...minerr ( ).*** Так как система нелинейных уравнений может иметь несколько решений, то полученные результаты зависят от начальных значений искомых переменных. В обоих случаях получаются приближенные решения, для которых рекомендуется делать проверку. Обычно требуется, чтобы количество уравнений было равно количеству искомых переменных, но в некоторых случаях, когда с точки зрения классической математики может быть получено точное решение и при меньшем количестве уравнений, данное условие может быть нарушено. На Листинге 2.17 представлены примеры использования блоков given...find( ) и given...minerr ( ) для решения систем нелинейных уравнений.

Листинг 2.17. **Результаты решения системы нелинейных уравнений**

Решение системы нелинейных уравнений с использованием блоков given...find( ) и given...minerr ( )

$y := 1 \quad x := 1 \quad \text{Given} \quad (x-2)^2 + (y-3)^2 = 0 \quad \text{Find}(x,y) = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$

$z := 0 \quad p := 0 \quad \text{Given} \quad (z-4)^2 + (p-5)^2 = 0 \quad \text{MinErr}(z,p) = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$

$m := 1 \quad n := 0 \quad \text{Given} \quad m^n + n^2 = 10 \quad m^2 + n^m = 2 \quad \text{MinErr}(m,n) = \begin{pmatrix} 0.492 \\ 3.145 \end{pmatrix}$

Проверка $\quad m := 0.492 \quad n := 3.145 \quad m^n + n^2 = 9.998 \quad m^2 + n^m = 1.999$

$p := 1 \quad q := 0 \quad \text{Given} \quad p^q + q^2 = 10 \quad p^2 + q^p = 2 \quad \text{Find}(p,q) = \begin{pmatrix} 0.492 \\ 3.145 \end{pmatrix}$

При изменении начальных условий функция minerr ( ) дает неверный результат, а функция find( ) не находит решения

$m := 0 \quad n := 0 \quad \text{Given} \quad m^n + n^2 = 10 \quad m^2 + n^m = 2 \quad \text{MinErr}(m,n) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

2.5.4. ПОИСК ЭКСТРЕМУМОВ ФУНКЦИЙ

Для поиска экстремумов в Mathcad существует две функции:

- ***Minimize (y,x)*** - для отыскания значения x, соответствующего локальному минимуму функции ***y(x);***
- ***Maximize (y,x)*** - для отыскания значения x, соответствующего локальному максимуму функции ***y(x).***

Так как ***y(x)*** может иметь несколько локальных экстремумов, а функции ***Minimize (y,x)*** и ***Maximize (y,x)*** позволяет найти только одно значение, то дополнительно задается начальное приближение переменной *x*. В результате находится значение экстремума функции *y(x)*, ближайшее к заданному начальному приближению переменной *x*.

При отыскании экстремумов посредством записи неравенства может быть задана некоторая область, в границах которой производится отыскание экстремума. В этом случае записи неравенства предшествует введение ключевого слова given. На листинге 2.18 представлен пример отыскания экстремумов функции в MathCad.

К сожалению, функция не всегда дает корректные результаты. Например, в примере на листинге 2.17 при поиске максимума в диапазоне -2<x<0 функция дает результат – 0.452. В то же время в другом примере она дает совсем иной результат:

$x := 1 \quad \text{Given} \quad -2 < x < 0 \quad \text{Maximize}(y,x) = 1.162$

В этом можно убедиться и по графику функции на листинге 2.18.

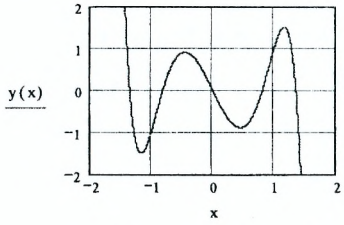
Отсюда можно сделать вывод, что результаты вычислений функций Minimize и Maximize могут зависеть от предшествующих вычислений. И поэтому применять данные функции надо с осторожностью, сверяя результаты по графикам функции.

Листинг 2.18. **Поиск локальных экстремумов функций**

Отыскание экстремумов функции

для функции $y(x) := -x^7 + 5 \cdot x^3 - 3 \cdot x$

$x := 1$



Minimize(y, x) = 0.452

Maximize(y, x) = 1.162

$x := -1$

Minimize(y, x) = −1.162

Maximize(y, x) = −0.452

$x := 1$ Given $-2 < x < 0$ Maximize(y, x) = −0.452

$x := 1$ Given $-2 < x < 0$ Minimize(y, x) = −1.162

2.5.5. ОПРЕДЕЛЕНИЕ ЧИСЛЕННОГО ЗНАЧЕНИЯ ПРОИЗВОДНОЙ

Численное значение производной может быть получено посредством указания конкретного числового значения переменной дифференцирования с последующим выполнением операции символьного дифференцирования и расчетом значения производной в указанной точке. Другой способ предусматривает первоначальное определение функции пользователя как производной от заданной функции, с последующим вычислением числового значения функции пользователя при заданном значении числового аргумента. Символьное дифференцирование было рассмотрено ранее в 2.3., поэтому в данном разделе не рассматривается. При расчетах следует помнить, что нельзя вставлять числовое значение переменной дифференцирования непосредственно в выражение дифференцирования, так как в этом случае дифференцируемая функция обращается в константу, производная которой будет равна 0. Примеры определения численного значения производной в MathCad приведены на листинге 2.19.

Листинг 2.19. **Пример расчета численного значения производной**

Расчет численного значения производной

Расчет посредством символьного дифференцирования

$x := 6$

$\frac{d}{dx} e^{x+1} \cdot \sin(x) \rightarrow \exp(7) \cdot \sin(6) + \exp(7) \cdot \cos(6) = 746.538$

Расчет посредством функции пользователя

$d(x) := \frac{d}{dx} e^{x+1} \cdot \sin(x) \rightarrow \exp(x+1) \cdot \sin(x) + \exp(x+1) \cdot \cos(x)$

$d(6) = 746.538$ $d(\pi) = -62.903$

Пример неверного расчета контрольного примера

$\frac{d}{dx} e^{6+1} \cdot \sin(6) \rightarrow 0$ $\frac{d}{dx} e^{6+1} \cdot \sin(6) = 0$

2.5.6. ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННОГО ИНТЕГРАЛА

Определенный интеграла в MathCad можно вычислить с помощью оператора интегрирования $\int_{\blacksquare}^{\blacksquare}$ на панели инструментов "Исчисление". В результате активизируется шаблон определенного интеграла, который имеет четыре поля для ввода пределов интегрирования, переменной интегрирования и функции интегрирования. Результат вычисления определенного интеграла отображается после ввода знака равно "=". При вычислении определенного интеграла для получения более точного и достоверного результата можно выбрать метод расчета путем вызова контекстного меню щелчком правой кнопки мыши по полю шаблона. По умолчанию активизирован режим "автовыбор", при котором MathCad автоматически выбирает один из методов.

$\int_{\blacksquare}^{\blacksquare} \blacksquare\, d\blacksquare$

Метод Romberg (Ромберг) используется как общий метод для гладких функций; Adaptive (адаптивный) – для функций быстроизменяющихся на участке интегрирования; Infinie Limit (неограниченный лимит) – бесконечный предел – для расчета интегралов с бесконечными пределами интегрирования. Метод Singular Endpoint (сингулярный конец) применяется для расчета интегралов от функций, не существующих в одной или обеих точках пределов интегрирования (так называемые несобственные интегралы второго рода).

При необходимости пользователь может отказаться от режима "автовыбор" и принудительно указать нужный метод расчета. На листинге 2.20 приведены примеры вычисления определенного интеграла. Как показывает анализ результатов расчета, режим автовыбор является наиболее универсальным при расчете определенного интеграла.

Листинг 2.20. **Примеры вычисления определенного интеграла**

Неограниченный лимит (автовыбор)

$$\int_1^{\infty} \frac{e^{-x}}{x^2}\,dx = 0.148$$

Ромберг и другие методы (не вычисляют)

$$\int_1^{\infty} \frac{e^{-x}}{x^2}\,dx = \blacksquare$$

Адаптивный (автовыбор), Ромберг, Сингулярный конец

$$\int_0^1 \frac{\sin(x)}{x}\,dx = 0.946$$

Неограниченный лимит(не вычисляет)

$$\int_0^1 \frac{\sin(x)}{x}\,dx = \blacksquare$$

Вычисляют все методы, кроме "Неограниченный лимит"

$$\int_{-2}^{2} x^3 - 3\cdot x^2 + 4\cdot x\,dx = -16$$

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимается под табулированием функций?
2. Для чего предназначена функция root ( )?
3. В чем отличие результатов вычисления корней полинома, получаемых при помощи функции root ( ) и polyroots ( )?
4. В чем специфика результатов решений уравнений, получаемых при помощи блока given...minerr ( )?
5. Какое значение позволяет вывести системная переменная ERR?
6. Для каких вычислений используется функция lsolve( )?
7. Для чего используются функции Minimize(y,x) и Maximize (y,x)?

2.6. МАТРИЧНЫЕ ВЫЧИСЛЕНИЯ

Создание векторов и матриц реализовано в MathCad различными способами, которые были рассмотрены в 2.2.5. В данном разделе рассматриваются только функции и операции с векторами и матрицами, а также прикладное использование векторов и матриц для решения некоторых задач.

2.6.1. ОСНОВНЫЕ ОПЕРАЦИИ С МАТРИЦАМИ И ВЕКТОРАМИ

Операции, выполняемые над векторами и матрицами, можно разбить на две большие группы. К первой группе относятся операции, которые применяются к отдельным векторам и матрицами. Например, транспонирование матрицы или вычисление обратной матрицы. Ко второй группе относятся операции, которые выполняются над группой векторов и матриц. Как правило, они выполняются над двумя матрицами или матрицей и вектором. Например, сложение, вычитание матриц, перемножение матриц или умножение матрицы и вектора. К векторам и матрицам, при выполнении операций над ними,

Таблица 2.5 Операции над векторами и матрицами MathCad

Операция	*Обозначение*	*Способ ввода*	*Описание*
Изменение знака	$-A$	-A	Умножает каждый элемент массива A на -1
Умножение массива на скаляр	$A*z$	A*z *или* z*A	Умножает каждый элемент массива A на скаляр z
Сложение массивов	$A1+A2$	A1+A2	Элементы массива A1 суммируются с соответствующими элементами A2
Матричное умножение	$A1 \cdot A2$	A1*A2	Возвращает произведение массива A1 на массив A2. Число столбцов A1 должно быть равно числу строк A2
Деление массива на скаляр	A/z	$\frac{A}{z}$	Делит каждый элемент массива A на скаляр z
Обращение матрица M	M^{-1}	M^-1	Находится матрица, обратная заданной.
Возведение матрицы в степень n	M^n	M^n	n раз перемножается матрица M посредством матричного умножения.
Определитель матрицы M	$\lvert M \rvert$	| M |	Рассчитывается определитель квадратной матрицы M, результат-скаляр
Транспонирование	A^T	A^T	Транспонирует массив A, т.е. меняет местами строки со столбцами
Выделение n-го столбца матрицы	$M^{<n>}$	M +CTRL ^n	Возвращает вектор в виде n – го столбца матрицы M
Выделение i, j – го элемента матрицы	$M_{i,j}$	M[i,j	Возвращает элемент матрицы i-ой строки j-го столбца

могут предъявляться определенные требования в соответствии с требованиями классической математики. Например, при перемножении матрицы и вектора, количество столбцов матрицы должно быть равно количеству срок вектора. Поэтому при работе с векторами и матрицами пользователь должен иметь необходимую математическую подготовку. В таблице 2.5. приведены основные операции, выполняемые над векторами и матрицами, используемые в MathCad. В таблице приняты следующие обозначение: A – массив, под которым понимается вектор или матрица, M – матрица, z - скаляр, v – вектор.

Наиболее удобно выполнять матричные вычисления с использованием кнопок панели инструментов "Матрица". По умолчанию индексация строк и столбцов элементов матрицы начинается с 0. Для того чтобы индексация начиналась с 1, необходимо системной переменной ORIGIN присвоить значение 1 (см. 2.2., 2.5.). На листинге 2.21 приведен пример матричных вычислений в MathCad.

Листинг 2.21. **Примеры операций с матрицами**

Матричные вычисления

Создание матрицы A и B

$$A := \begin{pmatrix} 3 & 6 & -3 \\ 4 & -2 & -5 \\ 2 & -5 & 8 \end{pmatrix} \qquad B := \begin{pmatrix} -2 & 3 & 7 \\ 3 & 5 & 9 \\ 1 & 2 & 6 \end{pmatrix}$$

Создание вектора v

$$v := \begin{pmatrix} 2 \\ -7 \\ 9 \end{pmatrix}$$

Сложение матриц A и B

$$A + B = \begin{pmatrix} 1 & 9 & 4 \\ 7 & 3 & 4 \\ 3 & -3 & 14 \end{pmatrix}$$

Матричное умножение A*B

$$A \cdot B = \begin{pmatrix} 9 & 33 & 57 \\ -19 & -8 & -20 \\ -11 & -3 & 17 \end{pmatrix}$$

Матричное умножение A*v

$$B \cdot v = \begin{pmatrix} 38 \\ 52 \\ 42 \end{pmatrix}$$

Расчет обратной матрицы A

$$A^{-1} = \begin{pmatrix} 0.125 & 0.101 & 0.11 \\ 0.128 & -0.092 & -9.174 \times 10^{-3} \\ 0.049 & -0.083 & 0.092 \end{pmatrix}$$

Определитель матрицы A

$$|A| = -327$$

Выделение 2 столбца матрицы A

$$A^{\langle 2 \rangle} = \begin{pmatrix} -3 \\ -5 \\ 8 \end{pmatrix}$$

Транспонирование матрицы B

$$B^{T} = \begin{pmatrix} -2 & 3 & 1 \\ 3 & 5 & 2 \\ 7 & 9 & 6 \end{pmatrix}$$

Матричное умножение A*A

$$A^{2} = \begin{pmatrix} 27 & 21 & -63 \\ -6 & 53 & -42 \\ 2 & -18 & 83 \end{pmatrix}$$

Матричное умножение обратных матриц

$$A^{-2} = \begin{pmatrix} 0.034 & -5.695 \times 10^{-3} & 0.023 \\ 3.872 \times 10^{-3} & 0.022 & 0.014 \\ 1.87 \times 10^{-5} & 4.938 \times 10^{-3} & 0.015 \end{pmatrix}$$

Выражение A $^{-2}$ отображает матричное умножение обратной матрицы A саму на себя. Выражение A $^{-n}$ соответственно отображает последовательное перемножение обратной матрицы A саму на себя n раз

2.6.2. МАТРИЧНЫЕ ФУНКЦИИ

MathCad имеет более 50 функций, предназначенных для работы с векторами и матрицами. Все функции можно разбить на группы по их функциональному назначению. Например, функции, предназначенные для создания матриц общего и специального вида, редактирования и преобразования матриц, функции, определяющие параметры матриц и т. д. Рассмотрим часть этих функций, которые имеют наибольшее прикладное значение.

Среди функций, предназначенных для создания матриц, следует выделить функцию ***matrix(L,N,f)***, где *L* – число строк матрицы, *N* – число столбцов матрицы, f – функция *f(l,n)* при $l = \overline{1, L};\ n = \overline{1, N}$. Другая функция из этой группы ***identity(n).*** Функция предназначена для создания единичной матрицы размерности n. Следующая функция ***geninv(M)*** позволяет осуществить обращение матрицы *M*, аналогично операции M^{-1}.

Листинг 2.22. **Примеры использования матричных функций**

Применение матричных функций

Создание матриц A и B

$f1(x,y) := -2 \cdot x + 1 + y^2$

$f2(x,y) := x^2 + 3 - y$

$A := \text{matrix}(3,3,f1)$ $B := \text{matrix}(3,3,f2)$ $A = \begin{pmatrix} 1 & 2 & 5 \\ -1 & 0 & 3 \\ -3 & -2 & 1 \end{pmatrix}$ $B = \begin{pmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 7 & 6 & 5 \end{pmatrix}$

Создание единичной матрицы

$C := \text{identity}(3)$ $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Вычисление обратной матрицы

$\text{geninv}(A) = \begin{pmatrix} 0.051 & -0.064 & -0.179 \\ 0.071 & -0.026 & -0.122 \\ 0.128 & 0.09 & 0.051 \end{pmatrix}$

Определение количество строк и столбцов матрицы

$\text{rows}(B) = 3$ $\text{cols}(C) = 3$

Сортировка матриц по возрастанию по строкам и столбцам

$\text{csort}(A,0) = \begin{pmatrix} -3 & -2 & 1 \\ -1 & 0 & 3 \\ 1 & 2 & 5 \end{pmatrix}$ $\text{rsort}(B,1) = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 5 & 6 & 7 \end{pmatrix}$

Выделение подматрицы

$\text{submatrix}(A,0,1,0,1) = \begin{pmatrix} 1 & 2 \\ -1 & 0 \end{pmatrix}$

Выделение вектора

$\text{submatrix}(B,0,2,1,1) = \begin{pmatrix} 2 \\ 3 \\ 6 \end{pmatrix}$

Слияние матриц

$P := \text{augment}(A,B)$

$P = \begin{pmatrix} 1 & 2 & 5 & 3 & 2 & 1 \\ -1 & 0 & 3 & 4 & 3 & 2 \\ -3 & -2 & 1 & 7 & 6 & 5 \end{pmatrix}$

$\text{stack}(B,A) = \begin{pmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ 7 & 6 & 5 \\ 1 & 2 & 5 \\ -1 & 0 & 3 \\ -3 & -2 & 1 \end{pmatrix}$

Для определения размерности матрицы предназначены функция ***rows(M)***, определяющая число строк матрицы *M*, и функция ***cols(M),*** определяющая число колонок матрицы *M*.

Сортировку элементов матрицы осуществляют две функции ***csort(M,i), rsort(M,j)***. Функция ***csort(M,i)*** обеспечивает сортировку по возрастанию элементов *i* – го столбца путем перестановки строк, а функция ***rsort(M,j)*** – сортировку по возрастанию элементов *j* –ой строки путем перестановки столбцов.

Для определения минимального и максимального элемента матрицы используются функции ***min(M) и max(M).***

Выделить произвольную подматрицу из матрицы M можно посредством функции ***submatrix (M, r1, r2, c1, c2),*** где *M* – исходная матрица, *r1 и r2* –нижний и верхний номер строки матрицы *M*, включаемых в результирующую подматрицу, а *c1 и c2* – нижней и верхний номер столбца матрицы *M*, включаемых в результирующую подматрицу. Слияние матриц можно осуществить, используя функции ***augment(A,B,...)*** и ***stack(A,B,...).*** Функция ***augment(A,B,...)*** предназначена для слияния матриц *A, B* и т.д. слева направо. Причем количество строк в матрицах должно быть одинаково. Вторая функция ***stack(A,B,...)*** выполняет слияние матриц сверху вниз. Количество столбцов в матрицах должно быть также одинаково. Данные функции могут быть применены и к векторам. На листинге 2.22. приведен пример использования рассмотренных матричных функций.

2.6.3. РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МАТРИЧНЫМ СПОСОБОМ

Рассмотрим системы линейных алгебраических уравнений в векторно-матричной форме ***A*x =b***, где ***A*** – квадратная матрица коэффициентов при неизвестных, причем определитель матрицы должен быть отличным от нуля; ***x-*** вектор неизвестных; ***b*** - вектор свободных членов. Решение данной сводится к следующему. Если определитель матрицы ***A*** отличен от нуля, то матрица ***A*** обратима. Тогда, умножив левую и правую часть и сходного уравнения на обратную матрицу (A^{-1}), получаем решение в виде $x=A^{-1}*b$. Реализовать полученное решение средствами Mathcad не представляет сложностей. Для случая бесконечного множества решений получаем сингулярную матрицу, Mathcad выдает сообщение *"Matrix is singular. Cannot compute its inversy – Матрица сингулярная. Нельзя вычислить эту инверсию"* и прерывает вычисления. На листинге 2.23 представлен пример решения систем линейных алгебраических уравнений.

Листинг 2.23. **Решение системы линейных алгебраических уравнений матричным способом**

$$x + 3 \cdot y = 5 \qquad 3 \cdot x - 5 \cdot y = 1$$

$$A := \begin{pmatrix} 1 & 3 \\ 3 & -5 \end{pmatrix} \qquad b := \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

$$A^{-1} = \begin{pmatrix} 0.357 & 0.214 \\ 0.214 & -0.071 \end{pmatrix} \qquad z := A^{-1} \cdot b \qquad z = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

решение
x=2 y=1

2.6.4. РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МЕТОДОМ КРАМЕРА

Метод Крамера также предназначен для решения системы n линейных уравнений с n неизвестными вида ***A*x =b*** (см. предыдущий раздел) на основе предварительного вычисления определителей системы, при условии, что определитель (Δ) матрицы *A* отличен от нуля. Метод основан на формулах Крамера, вида $x_i = \frac{\Delta_i}{\Delta}$, где Δ_i – определитель матрицы, полученной из матрицы A системы заменой i –го столбца, т.е. столбца коэффициентов при неизвестном x_i вектором свободных членов, т.е. b. На листинге 2.24. приведен пример решения системы линейных алгебраических уравнений методом Крамера.

Листинг 2.24. **Решение системы линейных алгебраических уравнений методом Крамера**

$$x + 3 \cdot y = 5 \qquad 3 \cdot x - 5 \cdot y = 1$$

$$A := \begin{pmatrix} 1 & 3 \\ 3 & -5 \end{pmatrix} \qquad b := \begin{pmatrix} 5 \\ 1 \end{pmatrix} \qquad \Delta := |A|$$

Выделение столбцов матрицы

$$s1 := \text{submatrix}(A, 0, 1, 0, 0) \qquad s2 := \text{submatrix}(A, 0, 1, 1, 1)$$

$$s1 = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \qquad s2 = \begin{pmatrix} 3 \\ -5 \end{pmatrix}$$

Формирование дополнительной i-ой матрицы путем слияния

$$A1 := \text{augment}(b, s2) \qquad A2 := \text{augment}(s1, b)$$

$$A1 = \begin{pmatrix} 5 & 3 \\ 1 & -5 \end{pmatrix} \qquad A2 = \begin{pmatrix} 1 & 5 \\ 3 & 1 \end{pmatrix}$$

Расчет определителей D1 и D2

$$\Delta 1 := |A1| \qquad \Delta 2 := |A2| \qquad \Delta = -14 \qquad \Delta 1 = -28 \qquad \Delta 2 = -14$$

$$x := \frac{\Delta 1}{\Delta} \qquad y := \frac{\Delta 2}{\Delta} \qquad x = 2 \qquad y = 1$$

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие операции над матрицей предопределяют следующие записи в Mathcad: | M |; M^{-1}; M^T; $M^{<2>}$, где M – матрица.
2. С какого значения по умолчанию начинается индексация строк и столбцов элементов матрицы?
3. Какая функция предназначена для создания матрицы?
4. С помощью какой функции можно создать единичную матрицу?
5. Каким двумя способами можно вычислить обратную матрицу?
6. Для чего предназначена функция submatrix (M, r1, r2, c1, c2)?
7. Какие функции осуществляют слияние матриц?

2.7. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

2.7.1. ФУНКЦИИ, ПРЕДНАЗНАЧЕННЫЕ ДЛЯ РЕШЕНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Для решения обыкновенных дифференциальных уравнений и систем обыкновенных дифференциальных уравнений в MathCad введен ряд функций. Рассмотрим их:

odesolve(x,b,step) - используется для решения обыкновенного дифференциального уравнения, заданного как в виде задачи Коши, так и в виде краевой задачи. Начальные условия и дифференциальное уравнение должны быть определены в блоке *given*. Параметры функции: *x* –переменная, по которой производится интегрирование; *b* - конечное значение промежутка решения; *step* – величина шага численного метода (параметр необязательный).

rkfixed(u,a,b,N,D) – реализует численное решение задачи Коши по методу Рунге – Кутта с фиксированным шагом. Имеет следующие преимущества перед ***odesolve(x,b,step)***: может быть использована в программных модулях и позволяет оперативно пересчитывать результаты при изменении параметров. Параметры функции: u-вектор начальных условий; *a u b* – граничные значения отрезка решения задачи; *N* – число интервалов разбиения отрезка *[a,b]*, *D(x,y)* –вектор-функция, содержащая правые части первых производных, записанные в символьном виде.

Rkadapt(u, a,b, N, D) - возвращает матрицу, содержащую таблицу значений решения задачи Коши на интервале от *a* до *b* для уравнения или системы обыкновенных дифференциальных уравнений, вычисленную методом Рунге-Кутта с переменным шагом и начальными условиями в векторе u, *D(x,y)* –вектор функция, содержащая правые части первых производных, записанная в символьном виде, *n* - число шагов.

Функция *Rkadapt ( )* вследствие автоматического подбора шага, как правило, дает более точный результат по сравнению с другими функциями.

2.7.2. РЕШЕНИЕ ЗАДАЧИ КОШИ

Задача Коши для дифференциальных уравнений n-го порядка с одной неизвестной (обыкновенное дифференциальное уравнение - ОДУ) формулируется следующим образом. Найти решение дифференциального уравнения

$$y^{(n)} = f(x, y, y', \ldots, y^{(n-1)})$$

в виде функции y=y(x), которая удовлетворяет заданным начальным условиям

$$y(x_0) = y_0,\ \ y'(x_0) = y'_0, \ldots, y^{(n-1)}(x_0) = y_0^{(n-1)},$$

где $y_0, y'_0, \ldots, y^{(n-1)}{}_0$ - заданные значение. Решение задачи Коши для обыкновенных дифференциальных уравнений второго и более высоких порядков можно свести к системе уравнений. Решение задачи Коши для ОДУ первого порядка в Mathcad с использованием различных функций приведено на листинге 2.25.

Наибольшее распространение для решения задачи Коши получил метод Рунге – Кутта. Суть метода состоит в последовательном отыскании искомого значения функции y_{i+1} по формуле

$y_{i+1} = y_i + \Delta y_i$, где

$$\Delta y_i = (k_1^i + 2k_2^i + 2k_3^i + k_4^i)/6$$

Листинг 2.25. **Решение обыкновенного дифференциального уравнения с начальными условиями (задачи Коши)**

Решение задачи Коши

Условие задачи $\left(\frac{d}{dx}y\right)\cdot x = 2\cdot\left(y + x^4\right)$ $y(1) = 0$ участок решения [1,5]

Точное аналитическое решение y3(z) $y3(z) := z^4 - z^2$

Решение с помощью функции rkfixed

$D(x,y) := 2\cdot\frac{\left(y + x^4\right)}{x}$ $y0 := 0$ $y2 := rkfixed(y0, 1, 5, 8, D)$

Решение с помощью блока Given/ odesolve

Given

$y4(1) = 0$ $\left(\frac{d}{dx}y4(x)\right) = 2\cdot\frac{\left(y4(x) + x^4\right)}{x}$

$y5 := odesolve(x, 5, 8)$

Решение с помощью функции rkadapt

$y6 := Rkadapt(y0, 1, 5, 8, D)$

$x3 := 1, 1.5 .. 5$ Сравнение результатов расчетов

y2 =

1	0
1.5	2.8
2	11.966
2.5	32.748
3	71.898
3.5	137.666
4	239.801
4.5	389.554
5	599.674

y5(x3) =

0
2.8
11.966
32.748
71.898
137.666
239.801
389.554
599.674

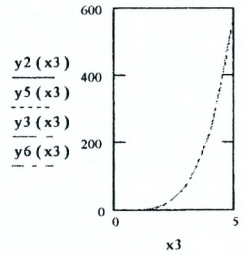
y6 =

1	0
1.5	2.812
2	12
2.5	32.812
3	72
3.5	137.812
4	240
4.5	389.812
5	600

y3(x3) =

0
2.813
12
32.813
72
137.813
240
389.813
600

Графическое отображение



при

$k_1^i = h * f(x_i, y_i)$

$k_2^i = h * f(x_i + h/2, y_i + k_1^i/2)$

$k_3^i = h * f(x_i + h/2, y_i + k_2^1/2)$

$k_4^i = h * f(x_i + h, y_i + k_3^i)$

$x_i = x_0 + i*h$ и $y_i = y(x_i)$, i=0,1

За *h* принимается достаточно малый шаг, с помощью которого весь интервал задачи Коши разбивается на дискретные точки, в которых и ищется решение. Погрешность результатов пропорциональна пятой степени шага (h^5).

Геометрический смысл метода Рунге – Кутта состоит в следующем. Из очередной точки (x_i,y_i) выбирается направление (угол) α_1, для которого $tg(\alpha_1)=f(x_i, y_i)$. На этом на-

правлении вычисляется точка с координатами $(x_i + \frac{h}{2}, y_i + \frac{k_1^i}{2})$. Затем из точки (x_i, y_i) выбирается направление (угол) α_2, для которого $tg(\alpha_2) = f(x_i + \frac{h}{2}, y_i + \frac{k_1^i}{2})$.

Листинг 2.26. **Решение краевой задачи**

Решение краевой задачи

Условие: Решить $y^V + y = 0$ На отрезке [0,1]

при граничных условиях $y(0) = 0$ y'(0)=7 y(1)=1 y'(1)=10 y''(1)=5

зададим начальные приближения для поиска величин y''(0) y'''(0) y''''(0)

x1 := 0 x2 := 1

$$v := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad load(x1, v) := \begin{pmatrix} 0 \\ 7 \\ v_0 \\ v_1 \\ v_2 \end{pmatrix} \qquad D(x1, y) := \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ -y_0 \end{pmatrix}$$

$$score(x2, y) := \begin{pmatrix} y_0 - 1 \\ y_1 - 10 \\ y_2 - 5 \end{pmatrix} \qquad v := sbval(v, 0, 1, D, load, score)$$

$$v = \begin{pmatrix} -85.014 \\ 348.107 \\ -516.257 \end{pmatrix} \qquad u := load(x1, v) \quad u = \begin{pmatrix} 0 \\ 7 \\ -85.014 \\ 348.107 \\ -516.257 \end{pmatrix}$$

Решение уравнение

r := rkfixed (u, 0, 1, 10, D)

r =

	x	y(x)	y'(x)	y''(x)	y'''(x)	y''''(x)
	0	1	2	3	4	5
0	0	0	7	-85.014	348.107	-516.257
1	0.1	0.331	0.153	-52.785	296.48	-516.28
2	0.2	0.129	-3.729	-25.718	244.851	-516.306
3	0.3	-0.333	-5.163	-3.815	193.221	-516.297
4	0.4	-0.839	-4.664	12.926	141.593	-516.238
5	0.5	-1.219	-2.75	24.504	89.975	-516.133
6	0.6	-1.359	0.065	30.921	38.368	-516.002
7	0.7	-1.193	3.263	32.178	-13.226	-515.872
8	0.8	-0.71	6.329	28.277	-64.808	-515.774
9	0.9	0.051	8.746	19.217	-116.383	-515.739
10	1	1	10	5	-167.959	-515.791

На этом направлении вычисляется точка с координатами $(x_i + \frac{h}{2}, y_i + \frac{k_2^i}{2})$. Далее из точки (x_i, y_i) выбирается направление (угол) α_3, для которого $tg(\alpha_3)=f(x_i + \frac{h}{2}, y_i + \frac{k_2^i}{2})$. На этом направлении вычисляется точка с координатами $(x_i + h, y_i + k_3^i)$. После чего из точки (x_i, y_i) выбирается направление (угол) α_4, для которого $tg(\alpha_4)=f(x_i + h, y_i + k_3^i)$. Все четыре полученных направления усредняются в соответствии с формулой для расчета Δy_i. На этом результирующем направлении и строится расчетная точка с координатами

$$(x_{i+1}, y_{i+1}) = (x_i + h, y_i + \Delta y_i)$$

Метод Рунге – Кутта благодаря высокой точности широко используется при численном решении дифференциальных уравнений и в частности в Mathcad. Существует несколько разновидностей данного метода, которые нашли свое отражение в рассмотренных выше функциях. На листинге 2.25 можно не только сравнить результаты, полученные на основе различных функций, но и оценить эти результаты с позиций точности расчетов.

Путем сравнения результатов решения задачи, можно сделать вывод о точности решения задачи. Наиболее точный результат позволяет получить функция *Rkadapt*.

2.7.3. КРАЕВЫЕ ЗАДАЧИ

Отличие краевой задачи от задачи Коши состоит в том, что в краевой задаче начальные условия задаются на концах интервала поиска решения. Для решения подобных задач в системе Mathcad используется метод пристрелки, который начальное условие в правой точке интервала преобразует в дополнительное начальное условие для левой точки интервала. После чего краевая задача трансформируется в задачу Коши, методы решения которой были рассмотрены в предыдущем разделе. Для реализации метода пристрелки в Mathcad существует функция ***sbval***. Данная функция определяет недостающие условия в начальной точке для двухточечных краевых задач. Функция имеет следующий синтаксис ***sbval(z,a,b,D,load,score),*** где *z* – вектор приближений недостающих начальных условий на левой границе; *a,b* – левая и правая граница интервала решений; *D(x,y)* – вектор-функция, содержащая правые части первых производных, записанная в символьном виде; ***load(a,z)*** – вектор-функция, описывающая начальные условия на левой границе интервала; ***score(b,y)*** – вектор-функция для задания правых граничных условий. Пример решения краевой задачи приведен на листинге 2.26.

2.7.4. МЕТОД КОНЕЧНЫХ РАЗНОСТЕЙ

В случае краевых задач для линейных дифференциальных уравнений применяются формулы для аппроксимации производных соответствующими конечно – разностными отношениями.

Это позволяет свести решение дифференциальных уравнений к решению системы линейных уравнений. Результаты получают в дискретных *i* – ых точках интервала решения задачи. При этом отрезок *[a,b]* разбивается на n частей с шагом *h =(b-a)/n*.

Для аппроксимации соответствующих производных используют следующие формулы:

$$y'_i = \frac{y_{i+1} - y_{i-1}}{2 * h}$$

$$y''_i = \frac{y_{i+1} - 2 * y_i + y_{i-1}}{h^2}, i = 1,2,...,n-1$$

Листинг 2.27. **Решения дифференциального уравнения методом конечных разностей**

Решение дифференциального уравнения методом конечных разностей

Условие : Найти решение y' ' + (1+x^2)*y=-1 на отрезке [-1;1]
при граничных условиях y(-1)=y(1)=0

$a := -1$ $b := 1$ Задаем количество интервалов $n := 8$

$h := \frac{(b-a)}{n}$ $h = 0.25$

$\frac{y_{k-1} - 2 \cdot y_k + y_{k+1}}{h^2} + \left[1 + \left(x_k\right)^2\right]^{-} \cdot y_k = -1$

$i := 0..8$ $x_i := -1 + h \cdot i$

$y_{k-1} - 2 \cdot y_k + y_{k+1} + \left[1 + \left(x_k\right)^2\right]^{-} \cdot y_k \cdot h^2 = -1 \cdot h^2$

приводим подобные и получаем конечно -разностную форму

$y_{k-1} + \left[\left[1 + \left(x_k\right)^2\right]^{-} \cdot h^2 - 2\right]^{-} \cdot y_k + y_{k+1} = -1 \cdot h^2$

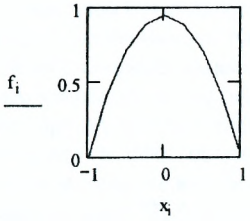
$k := 1..n-1$ $d_k := \left[1 + \left(x_k\right)^2\right]^{-} \cdot h^2 - 2$

Создадим матрицу коэффициентов при неизвестных для СЛАУ

$M_{k,k-1} := 1$ $M_{k,k+1} := 1$ $M_{k,k} := d_k$ $M_{0,0} := 1$ $M_{n,n} := 1$

Создадим вектор свободных членов $q_0 := 0$ $q_n := 0$ $q_k := -h^2$

Решим СЛАУ $f := \text{lsolve}(M, q)$



$$f = \begin{pmatrix} 0 \\ 0.4 \\ 0.699 \\ 0.881 \\ 0.942 \\ 0.881 \\ 0.699 \\ 0.4 \\ 0 \end{pmatrix}$$

Таким образом, сделав соответствующую замену, получаем систему линейных уравнений, решение которой средствами Mathcad не представляет сложностей. Решение задачи методом конечных разностей приведено на листинге 2.27.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение функции odesolve(x,b,ste,)?
2. Назначение функции rkfixed(u,a,b,N,D ?
3. Назначение функции Rkadapt(u, x1, x2, n, D)?
4. Назначение параметров функции sbval(z,a,b,D,load,score)?
5. Назначение функций load(a,z) и score(b,y)?

2.8. РЕШЕНИЕ ПРИКЛАДНЫХ ЗАДАЧ

2.8.1. ОБРАБОТКА ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

Интерполяция и экстраполяция данных

Под интерполяцией понимают восстановление функции по известным ее значениям или значениям ее производных в отдельных точках. Задача интерполяции экспериментальных данных сводится к тому, чтобы предсказать в промежуточных точках значение функции, заданной таблично. То есть исходные данные можно представить в виде таблицы, куда сводятся дискретные экспериментальные значения, полученные в некоторых точках наблюдений или в определенные интервалы времени. В Mathcad можно соединять табличные точки прямой линией (линейная интерполяция) либо отрезками кубического полинома (кубическая сплайн-интерполяция).

Линейная интерполяция реализуется посредством функции ***linterp(vx,vy,x),*** где *vx, vy* - векторы данных. Причём данные должны быть упорядочены по возрастанию; *x* - аргумент, для которого возвращается значение *y*

Кубическая сплайн-интерполяция позволяет провести через набор точек гладкую кривую так, чтобы в этих точках были непрерывны первая и вторая производные. Интерполяция осуществляется двумя функциями. Вначале вычисляется вектор вторых производных в рассматриваемых точках, затем вычисляется значение функции в точке ***x → interp(vs,vx,vy,x)***. Для построения вектора вторых производных в Mathcad имеется набор из 3-х функций, которые отличаются лишь граничными условиями: ***cspline(vx,vy)*** - генерирует кривую, являющуюся кубическим полиномом в граничных точках; ***pspline(vx,vy)*** - соответственно, параболу; ***lspline(vx,vy)*** - прямую.

На листинге 2.28 в качестве примера рассмотрена задача интерполяции значений прогиба балки по ее длине на основе шести исходных значений, полученных посредством уравнения строительной механики.

Под экстраполяцией понимают предсказание поведения функции за пределами области ее определения. В нашем случае эта задача сводится к определению значений некоторого параметра за пределами области, в которой значения этого параметра известны. ближайших к правой границе точек, на основе которых производится экстраполяция, *n* - количество точек, в которых производится экстраполяция данных. Результаты, получаемые на основе функции *predict(v,m,n)* в значительной мере зависят от параметра m. На листинге 2.29 представлены результаты работы функции на примере данных об объеме выпускаемой продукции за 12 месяцев.

Листинг 2.28. **Результаты интерполяции значений прогиба балки**

Интерполяция значений прогиба балки по ее длине

$F := 4.3$ $N := 6$ $i := 0, 1 .. 5$ $vx_i := \frac{400}{5} \cdot (i)$ $L := 400$ $a := 220$ $I := 800$ $E := 2 \cdot 10^6$

ЭТАЛОННЫЕ ЗНАЧЕНИЯ

$$vy_i := F \cdot \frac{(L-a) \cdot \left[\left(vx_i\right)^3 - a \cdot (2 \cdot L - a) \cdot vx_i - \text{if}\left[vx_i > a, L \cdot \left(vx_i - a\right)^3, 0 \right] \right]}{6 \cdot L \cdot E \cdot I}$$

i =	vx_i =	vy_i =
0	0	0
1	80	-0.00195
2	160	-0.00329
3	240	-0.00339
4	320	-0.00207
5	400	0

Например: Линейная интерполяция

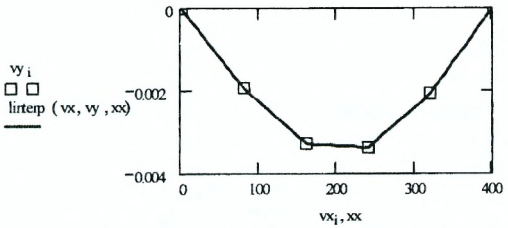
$x := 2$ $linterp(vx, vy, x) = -4.886 \times 10^{-5}$

$x := a$ $linterp(vx, vy, x) = -3.365 \times 10^{-3}$

$x := 333$ $linterp(vx, vy, x) = -1.736 \times 10^{-3}$

$xx := 0, 10 .. 400$

Построим график.



$vsc := cspline(vx, vy)$ $vsp := pspline(vx, vy)$ $vsl := lspline(vx, vy)$

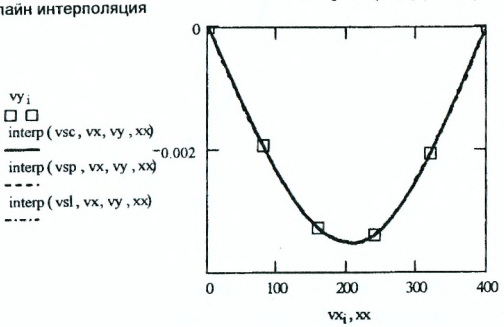
Сплайн интерполяция

$x := 380$

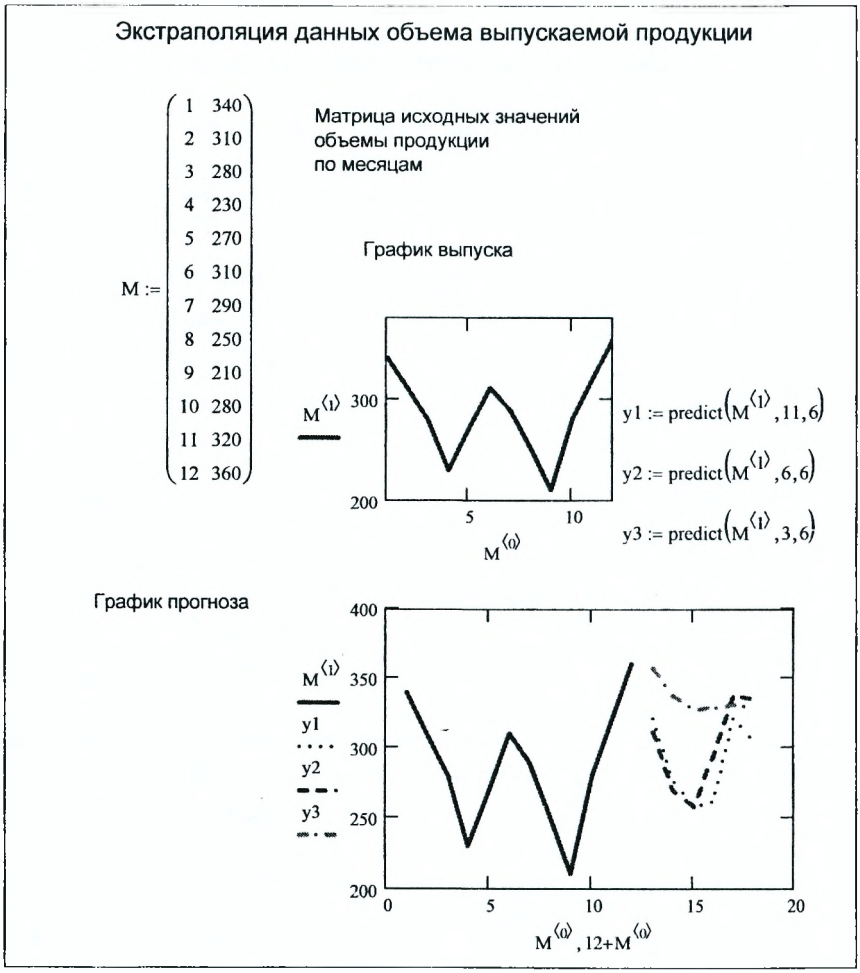
$interp(vsc, vx, vy, x) = -5.425 \times 10^{-4}$

$interp(vsp, vx, vy, x) = -5.725 \times 10^{-4}$

$interp(vsl, vx, vy, x) = -5.469 \times 10^{-4}$



Листинг 2.29. **Экстраполяция данных объема выпускаемой продукции**



Регрессионный анализ

При обработке экспериментальных данных с целью исследования их природы, возникает необходимость выразить зависимую переменную в виде некоторой математической функции от одной или нескольких независимых переменных. Данная зависимость получила название регрессионная модель или уравнение регрессии, а методы, позволяющие получить эту зависимость, принято называть методами регрессионного анализа. Методы регрессионного анализа позволяют: производить расчет различного вида регрессионных моделей; проверять гипотезу адекватности модели имеющимся наблю-

дениям; использовать модель для прогнозирования значений зависимой переменной при новых значениях независимой переменной. В Mathcad существует набор функций, позволяющих рассчитать различные регрессионные модели. В таблице 2.6. представлены функции, используемые при создании регрессионных моделей.

Таблица 2.6 Функции, используемые при создании регрессионных моделей

Наименование модели	*Вид уравнения регрессии*	Функция Mathcad
Линейная	$y(x) = a*x + b$	*line(vx , vy)*
Полиномиальная n-ой степени	$y(x) = \sum_{i=0}^{n} a_n * x^n$	*p1:=regress(vx,vy,n)* *interp(p1,vx,vy,x)*
Фрагменты полиномов 2-ой степени	$y_i(x) = \sum a_i * x^2 + b_i * x + c_i$	*p2:=loess(vx,vy,span)* *interp(p2,vx,vy,x)*
Экспоненциальная	$y(x)=a*e^{b*x}+c$	*expfit(vx,vy,g)*
Логистическая функция	$y(x)=a/(1+b*e^{-c*x})$	*lgsfit(vx,vy,g)*
Синусоидальная	$y(x)=a*sin(x+b)+c$	*sinfit(vx,vy,g)*
Степенная	$y(x)=a*x^b+c$	*pwfit(vx,vy,g)*
Логарифмическая	$y(x) = a*ln(x + b)+c$	*logfit(vx,vy,g)*
Логарифмическая короткая	$y(x) = a*ln(x)+b$	*lnfit(vx,vy,g)*

Рассмотрим суть параметров, используемых в качестве аргументов в функциях. В каждой функции используются два вектора исходных данных, vx - вектор независимых переменных, vy - вектор зависимых переменных. Количество элементов вектора vx и vy должно быть одинаково. Функции regress и loess используются только совместно с функцией interp. Сами функции regress и loess вычисляют только вектор, требуемый функцией interp для определения самого полинома. Параметр span функции loess определяет величину области, на которой строится конкретный фрагмент полинома 2-ой степени. Оптималь ное значение span, предлагаемое справочной системой Mathcad, равно 0.75, но в каждом конкретном случае рекомендуется путем вариантных расчетов подобрать наилучшее значение span. Параметр g является вектором начальных приближений для неизвестных функции регрессии. После определения регрессионных зависимостей, актуальным является выбор из их совокупности наилучшей функции, с точки зрения адекватности описания исходных экспериментальных данных. В качестве критерия, позволяющего выбрать наилучшую регрессионную модель, предлагается использовать коэффициент детерминации, численно равный коэффициенту корреляции в квадрате. Значение коэффициента корреляции в MathCad позволяет рассчитать функция corr(A,B), где A и B – два вектора значений. На листинге 2.30, представлен пример расчета различных регрессионных моделей и выбора наилучшей из них.

Как видно из данных, приведенных на листинге 2.30 наилучшие результаты дает полиномиальная модель на основе функции loess. Данная модель характеризуется значением коэффициента детерминации равным 0.984.

Листинг 2.30. **Расчет регрессионных моделей строительства автодорог**

Регрессионная модель затрат (млн. ф.ст.) на строительство автодорог в Англии за 11 лет

$$vx := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{pmatrix} \quad vy := \begin{pmatrix} 560 \\ 608 \\ 685 \\ 839 \\ 814 \\ 914 \\ 1100 \\ 1196 \\ 1490 \\ 1574 \\ 1513 \end{pmatrix}$$

Исходные данные

начальные приближения неизвестных для регрессионных моделей

$$g := \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix}$$

линейная модель

$k1 := line(vx, vy)$

$y1(x) := k1_0 + k1_1 \cdot x$

$kd_1 := corr(vy, y1(vx))^2$

$kd_1 = 0.95$

полиномиальная модель

полином 3-ей степени

$p2 := regress(vx, vy, 3)$

$y2(x) := interp(p2, vx, vy, x)$

$kd_2 := corr\left(vy, \overrightarrow{y2(vx)}\right)^2$

$kd_2 = 0.969$

полиномиальная модель - loess

$p3 := loess(vx, vy, 0.75)$

$y3(x) := interp(p3, vx, vy, x)$

$kd_3 := corr\left(vy, \overrightarrow{y3(vx)}\right)^2$

$kd_3 = 0.984$

экспоненциальная модель

$p4 := expfit(vx, vy, g)$

$y4(x) := p4_0 \cdot e^{p4_1 \cdot x} + p4_2$

$kd_4 := corr\left(vy, \overrightarrow{y4(vx)}\right)^2$

$kd_4 = 0.959$

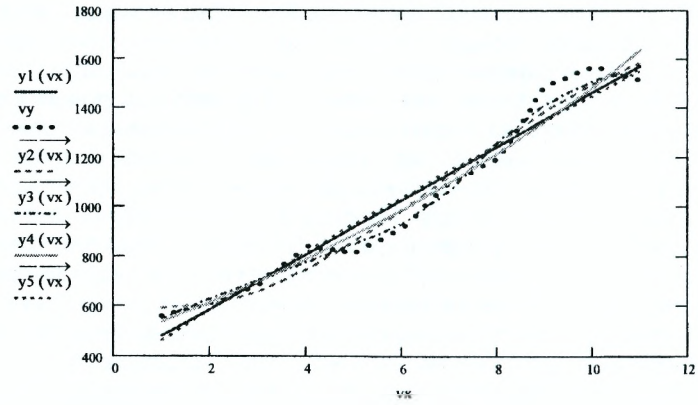
логарифмическая функция

$p5 := logfit(vx, vy, g)$

$y5(x) := p5_0 \cdot \ln(p5_1 + x) + p5_2$

$kd_5 := corr\left(vy, \overrightarrow{y5(vx)}\right)^2$

$kd_5 = 0.943$



2.8.2. РЕШЕНИЕ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

К классу задач линейного программирования относятся задачи, в которых требуется оптимизировать (определить максимум или минимум) целевую функцию вида $z = f(x_1, x_2, ..., x_n) = c_1 * x_1 + c_2 * x_2 + ... + c_n * x_n$

при следующих ограничениях

$$a_{i1} * x_1 + a_{i2} * x_2 + ... + a_{in} * x_n - b_i \geq 0, \quad (i = 1,2,...,m)$$

$$x_k \geq 0, \quad (k = 1,2,..n)$$

Одним из способов решение подобных задач в Mathcad является использование блока given с функциями minimize и maximize (см. 2.2.5.). В качестве примера рассмотрим задачу планирования производства красок. Суть задачи в следующем. Фабрика выпускает два типа красок - I и E. Для производства красок используются два компонента - A и B. Максимальные суточные запасы этих компонентов: компонент A - 6 тонн; компонент B – 8 тонн. Расходы компонентов A и B на производство 1 тонны краски следующие: для краски I - A/B= 2/1; для краски E - A/B = 1/2. Суточный спрос на краску I никогда не превышает спрос на краску E более, чем на 1 тонну. Спрос на краску I никогда не

Листинг 2.31. **Решение задачи линейного программирования**

Решение задачи линейного программирования.

ORIGIN := 1 Задание целевой функции

$F(x) := 2000 \cdot x_1 + 3000 \cdot x_2$

Задание начальных приближений $x := \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

Given

Ввод ограничений

$x_2 + 2 \cdot x_1 \leq 6$

$2 \cdot x_2 + x_1 \leq 8$

$x_1 - x_2 \leq 1$

$x_1 \leq 2$

$x_1 \geq 0$

$x_2 \geq 0$

Решение

$x := \text{Maximize}(F, x) \quad x = \begin{pmatrix} 1.333 \\ 3.333 \end{pmatrix}$

Значение функции цели $F(x) = 1.267 \times 10^4$

превышает 2 тонн в сутки. Оптовые цены на краску I 2000 рублей за тонну, а на краску E – 3000 рублей за тонну. Определить максимальный доход фабрики от продажи краски. Обозначив суточный объем выпуска краски I за x_1, а суточный объем выпуска краски E за x_2, получаем экономико-математическую модель задачи, вид которой и листинг ее решения в MathCad приведены на листинге 2.31.

2.8.3. РЕШЕНИЕ ЗАДАЧ МЕХАНИКИ

В качестве примера рассмотрим задачу строительной механики. Рассчитать прогиб однопролетной балки длиной *L*, закрепленной на концах, при приложении к ней сосредоточенной силы *F* на расстоянии *a* от левого конца балки (см. рис. 2.10). На данном примере рас-

сматривается возможность практического применения некоторых рассмотренных выше функций и методов MathCad. Рассмотрим метод символьного интегрирования. Основой метода является дифференциальное уравнение вида y"=M(x)/(E*I). Для участка слева от точки приложения силы F, M(x) вычисляется по формуле M(x)=F*(L-a)*x/L. Для участка справа от точки приложения силы F, M(x) вычисляется по формуле M(x)=F*((L-a)*x/L-(x-a)). Решение основано на двойном интегрировании и представлено на листинге 2.32. Дополнительно была сделана проверка результатов расчета путем сравнения с данными расчета, полученными на основе аналитического уравнения прогиба балки (листинг 2.32).

Листинг 2.32. **Расчет прогиба балки методом символьного интегрирования**

Расчет прогиба балки различными методами

Исходные данные $F := 4.3 \quad L := 400 \quad a := 220$

Жесткостные характеристики $I := 800 \quad E := 2 \cdot 10^6$

Метод символьного интегрирования

Основан на дифференциальном уравнении прогиба балки y"=M(x)/(E*I). Уравнение, описывающее M(x) слева от точки приложения силы F: M(x)=F*(L-a)*x/L. Уравнение, описывающее M(x) слева от точки приложения силы F: M(x)=F*((L-a)*x/L-(x-a)). Решение основано на двойном интегрировании.

Интегрируем функцию справа от точки приложения силы

$$\int \left[(L-a)\cdot\frac{x}{L}-(x-a)\right]\cdot F\,dx$$ В результате интегрирования получаем

$$y4(x,c1) := F\cdot\left[\frac{1}{2}\cdot(L-a)\cdot\frac{x^2}{L}-\frac{1}{2}\cdot x^2+a\cdot x\right]+c1$$ Константу c1 записываем принудительно

$$\int F\cdot\left[\frac{1}{2}\cdot(L-a)\cdot\frac{x^2}{L}-\frac{1}{2}\cdot x^2+a\cdot x\right]+c1\,dx$$ В результате повторного интегрирования получаем

$$y1(x,c1,c2) := F\cdot\left[\frac{1}{3}\cdot\left(\frac{1}{2}\cdot L-\frac{1}{2}\cdot a\right)\cdot\frac{x^3}{L}-\frac{1}{6}\cdot x^3+\frac{1}{2}\cdot a\cdot x^2\right]+c1\cdot x+c2$$ Константу c2 записываем принудительно

Интегрируем функцию слева от точки приложения силы

$$\int F\cdot(L-a)\cdot\frac{x}{L}\,dx$$ В результате интегрирования получаем

$$y3(x,c3) := F\cdot\left[\frac{1}{2}\cdot(L-a)\cdot\frac{x^2}{L}\right]+c3$$ Константу c3 записываем принудительно

$$\int F\cdot\left[\frac{1}{2}\cdot(L-a)\cdot\frac{x^2}{L}\right]+c3\,dx$$ В результате повторного интегрирования получаем

$$y2(x,c3,c4) := F\cdot\left[\frac{1}{3}\cdot\left(\frac{1}{2}\cdot L-\frac{1}{2}\cdot a\right)\cdot\frac{x^3}{L}\right]+c3\cdot x+c4$$ Константу c4 записываем принудительно

Обобщенное уравнение прогиба балки

$$y(x,c1,c2,c3,c4) := \frac{1}{E\cdot I}\cdot if(x<a,y2(x,c3,c4),y1(x,c1,c2))$$

Листинг 2.32. **Расчет прогиба балки методом символьного интегрирования (продолжение)**

Нахождение неизвестных

$c1 := 1 \quad c2 := 1 \quad c3 := 1 \quad c4 := 1$

Given

$y1(L, c1, c2) = 0$ -->Прогиб балки на правом конце = 0

$y2(0, c3, c4) = 0$ -->Прогиб балки на левом конце = 0

$y1(a, c1, c2) = y2(a, c3, c4)$ -->Прогиб балки в точке приложения силы слева и справа одинаков

$y3(a, c3) = y4(a, c1)$ -->В точке приложения силы функция прогиба не имеет разрыва

$$\text{Find}(c1, c2, c3, c4) = \begin{pmatrix} -1.452 \times 10^5 \\ 7.631 \times 10^6 \\ -4.115 \times 10^4 \\ 0 \end{pmatrix}$$

$c1 := -1.452 \cdot 10^5 \quad c2 := 7.631 \cdot 10^6$

$c3 := -4.115 \cdot 10^4 \quad c4 := 0$

Проверка

$j := 0, 1 .. 5 \quad t_j := 80 \cdot j$

$$yy_j := F \cdot \frac{(L - a) \cdot \left[(t_j)^3 - a \cdot (2 \cdot L - a) \cdot t_j \right] - \text{if}\left[t_j > a, L \cdot (t_j - a)^3, 0 \right]}{6 \cdot L \cdot E \cdot I}$$

$t_j =$

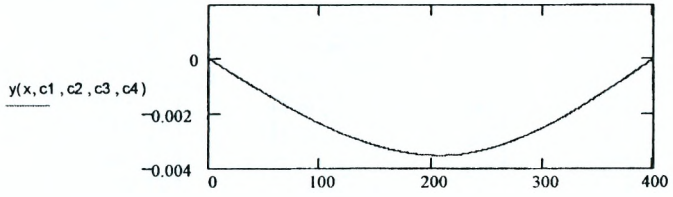
0
80
160
240
320
400

$y(t_j, c1, c2, c3, c4) =$

0
$-1.954 \cdot 10^{-3}$
$-3.289 \cdot 10^{-3}$
$-3.388 \cdot 10^{-3}$
$-2.071 \cdot 10^{-3}$
$2.708 \cdot 10^{-6}$

$yy_j =$

0
$-1.954 \cdot 10^{-3}$
$-3.289 \cdot 10^{-3}$
$-3.39 \cdot 10^{-3}$
$-2.073 \cdot 10^{-3}$
0



Листинг 2.33. **Расчет прогиба балки методом конечных разностей.**

Расчет прогиба балки методом конечных разностей

$F := 4.3 \quad a := 220 \quad E := 2 \cdot 10^6 \quad I := 800 \quad L := 400 \quad i := 0, 1 .. 10 \quad x_i := \frac{L}{10} \cdot i$

$M(x) := F \cdot (L - a) \cdot \frac{x}{L} - (x \geq a) \cdot F \cdot (x - a) \quad b(x) := \frac{M(x)}{E \cdot I} \cdot \left(\frac{L}{10}\right)^2$

$x_i =$	$M(x_i) =$	$b(x_i) =$
0	0	0
40	77.4	$7.74 \cdot 10^{-5}$
80	154.8	$1.548 \cdot 10^{-4}$
120	232.2	$2.322 \cdot 10^{-4}$
160	309.6	$3.096 \cdot 10^{-4}$
200	387	$3.87 \cdot 10^{-4}$
240	378.4	$3.784 \cdot 10^{-4}$
280	283.8	$2.838 \cdot 10^{-4}$
320	189.2	$1.892 \cdot 10^{-4}$
360	94.6	$9.46 \cdot 10^{-5}$
400	0	0

Формирование матрицы коэффициентов при неизвестных для СЛАУ метода конечных разностей

ORIGIN= 0 $\quad m := 0, 1 .. 10 \quad MA_{m,m} := 0$

$k := 1, 2 .. 9 \quad MA_{k,k} := -2 \quad MA_{k,k+1} := 1$

$MA_{k,k-1} := 1 \quad MA_{0,0} := 1 \quad MA_{10,10} := 1$

Формирование вектора свободных членов

$T_i := b(x_i)$

Решение СЛАУ $\quad kr := \text{lsolve}(MA, T)$

Проверка

$$y_i := F \cdot \frac{(L - a) \cdot \left[(x_i)^3 - a \cdot (2 \cdot L - a) \cdot x_i - \text{if}\left[x_i > a, L \cdot (x_i - a)^3, 0 \right] \right]}{6 \cdot L \cdot E \cdot I}$$

$i =$	$x_i =$	$y_i =$
0	0	0
1	40	$-1.016 \cdot 10^{-3}$
2	80	$-1.954 \cdot 10^{-3}$
3	120	$-2.738 \cdot 10^{-3}$
4	160	$-3.289 \cdot 10^{-3}$
5	200	$-3.531 \cdot 10^{-3}$
6	240	$-3.39 \cdot 10^{-3}$
7	280	$-2.873 \cdot 10^{-3}$
8	320	$-2.073 \cdot 10^{-3}$
9	360	$-1.084 \cdot 10^{-3}$
10	400	0

$kr =$

	0
0	0
1	$-1.019 \cdot 10^{-3}$
2	$-1.961 \cdot 10^{-3}$
3	$-2.748 \cdot 10^{-3}$
4	$-3.302 \cdot 10^{-3}$
5	$-3.548 \cdot 10^{-3}$
6	$-3.406 \cdot 10^{-3}$
7	$-2.885 \cdot 10^{-3}$
8	$-2.081 \cdot 10^{-3}$
9	$-1.088 \cdot 10^{-3}$
10	0

Рассмотрим применение метода конечных разностей для приведенной задачи. Алгоритм расчета методом конечных разностей аналогичен алгоритму, рассмотренному в 2.7.4, а результаты расчетов представлены на листинге 2.33.

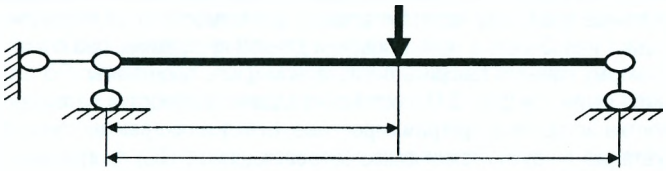


Рис 2.10. Схема однопролетной балки

ЗАКЛЮЧЕНИЕ

В данном разделе описаны возможности математической системы Mathcad по организации вычислений. Рассмотрены функции, реализующие символьные и числовые расчеты, матричные вычисления, решение дифференциальных уравнений, обработку данных. Приведены сведения о средствах Mathcad для отображения результатов вычислений в виде графиков и диаграмм. Раздел насыщен примерами, иллюстрирующими практическое использование приведенных теоретических сведений для решения инженерных задач, что показывает возможности прикладного использования Mathcad в различных областях и при изучении различных технических дисциплин.

3. АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

3.1. 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. 3.1): постановка задачи, разработка математической модели, разработка алгоритма, программирование, отладка программы, передача программы в эксплуатацию и научно-техническое сопровождение (НТС) программы, завершение жизненного цикла.

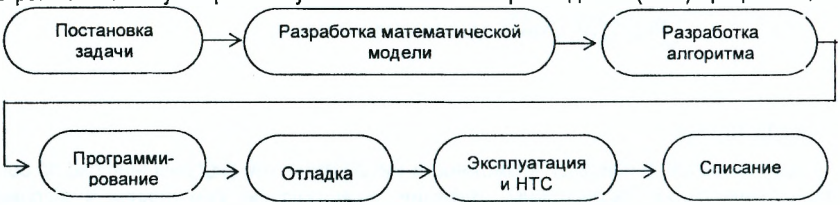


Рис. 3.1. Этапы жизненного цикла программы

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.

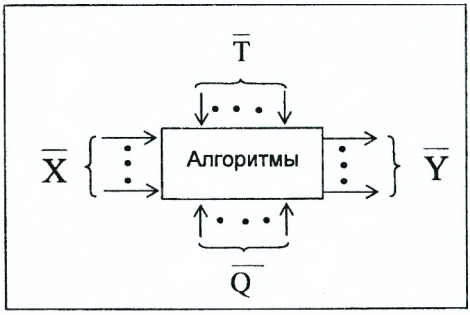


Рис. 3.2. Модель решения задачи

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. 3.2.), на вход которого поступают исходные данные – вектор $\overline{X}$, ограничения на входные параметры - век-

тор $\overline{Q}$, 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 значения, 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.

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 на ЭВМ.

Математическая модель задачи представляет собой совокупность математических выражений, описывающих данную задачу. В общем виде математическую модель можно представить следующим образом:

Функция_цели = $F(x_1, x_2, \ldots, x_n, q_1, q_2, \ldots, q_m, t_1, t_2, \ldots, t_p)$ (3.1)

min

Функциональные ограничения:

$$q_1 <= \varphi_1(x_1, x_2, \ldots x_n)$$
$$q_2 <= \varphi_2(x_1, x_2, \ldots x_n)$$
$$\ldots$$
$$q_m = \varphi_m(x_1, x_2, \ldots x_n)$$

Ограничения на значения параметров:

$$x_1 >= T_1$$
$$x_2 >= T_2$$
$$\ldots$$
$$x_n >= T_n$$

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. 3.3). В результате образуется *функциональная схема алгоритма.*

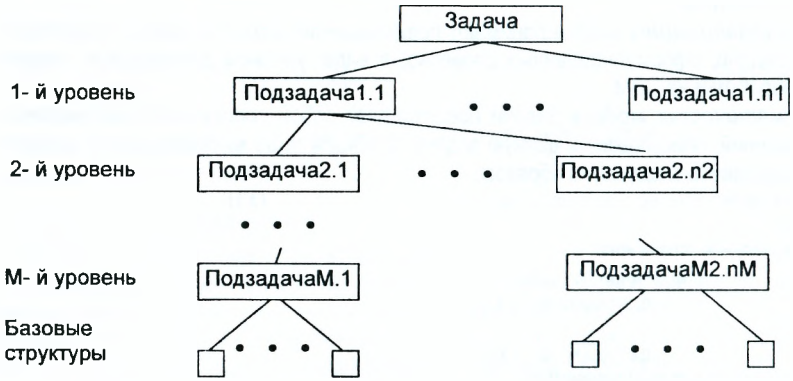


Рис. 3.3. К понятию "нисходящее проектирование"

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.

3.2. СХЕМА АЛГОРИТМА

3.2.1. ОПРЕДЕЛЕНИЕ И ОСНОВНЫЕ СВОЙСТВА

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 за конечное число шагов или заданное время. 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.

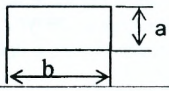
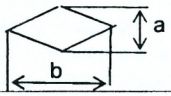
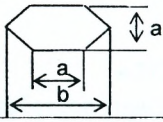
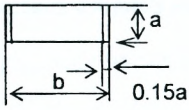
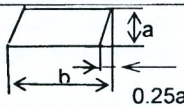
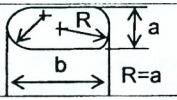
3.2.2. ПРЕДСТАВЛЕНИЕ АЛГОРИТМОВ

Для представления алгоритмов используются различные способы:

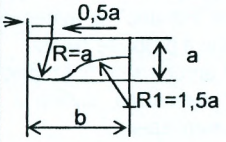
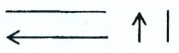
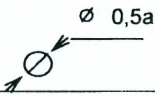
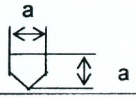
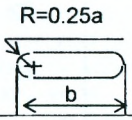
- словесный;
- на алгоритмическом языке;
- в виде схем алгоритмов;
- предикатная форма записи.

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.

Таблица 3.1 Основные графические элементы схем алгоритмов

Наименование	Обозначение	Функция
Процесс		Выполнение операций или группы операций, в результате которых изменяется значение, форма представления или расположение данных
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
Модификация		Выполнение операций, меняющих команды или группы команд, с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы)
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

Продолжение таблицы 3.1

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.
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

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. 3.2.

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. 3.1.

Размер стороны *a* выбирается из ряда 10, 15, 20, ... мм с шагом 5 мм. Размер стороны *b* принимается равным *1,5a*, допускается *2a*. ГОСТ не накладывает строгих ограничений на размеры элементов схем.

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.

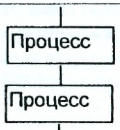
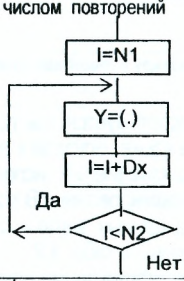
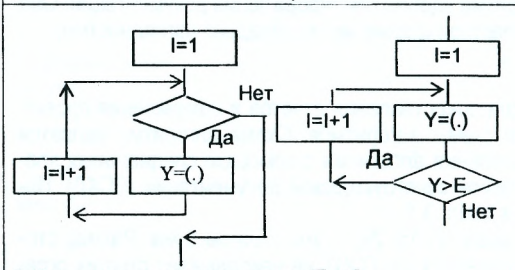
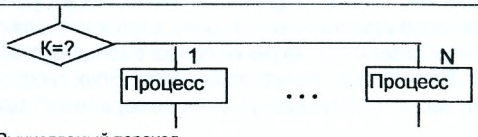
Блок "Начало" имеет только выход. Блок "Конец" имеет только вход. Блок процесса имеет один вход (сверху/снизу) и один выход (снизу/сверху). Блок выбора имеет один

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.

Таблица 3.2 Представление базовых структур алгоритмов

Схема алгоритма	Запись на алгоритмическом языке
А. Следование 	*Нач* <выражение> ... <выражение> *Кон*
Б. Решние 	*Если* <условие> *то* <выражение> *иначе* <выражение> *Все*
В. Цикл с заданным числом повторений (цикл "ДО") 	*Для* I от N1 до N2 шаг Dx *нц* <выражение> ... <выражение> *кц* --- For i=N1 TO N2 Step Dx < Тело цикла> Next i
 Г. Цикл с параметром или цикл типа "Пока" а) с предусловием б) с постусловием	*Пока* <условие> *нц* <выражение> ... <выражение> *кц* --- While <условие> <Тело цикла> Wend
 Д. Вычисляемый переход	*Выбор* при условии 1:<выраж.> ... при условии N:<выраж.> *Все*

Отличия могут заключаться только в форматах используемых операторов. Базовые структуры схем алгоритмов приведены в табл. 3.2.

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 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.3.2 д.

КОНТРОЛЬНЫЕ ВОПРОСЫ

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. Разработайте алгоритм для печати таблицы умножения.

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с основными свойствами алгоритма, способами его представления и базовыми структурами алгоритма. Таких структур четыре: линейная, выбор, цикл, вычисляемый переход. При выполнении схем алгоритмов следует руководствоваться требованиями ГОСТ на выполнение электронных схем.

3.3. ПРОГРАММИРОВАНИЕ

3.3.1. 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 (рис. 3.4).

Рис. 3.4. Схема алгоритма линейной программы

Операторы ввода данных

Для ввода данных используются операторы LET.

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.

Сущность выполнения оператора LET состоит в следующем: переменной, имя которой указано слева от знака "=", присваивается значение выражения, записанного справа от знака равно.

Примечание. Операторы, рассматриваемые в примерах относятся к языку программирования Visual Basic 6.0. Для проверки работы операторов или приводимых фрагментов программ необходимо запустить программу Visual Basic 6.0, открыть окно Программы (Code), поместить текст программы в обработчик события Click формы (или кнопки), запустить программу командой *Run, Start* и после появления формы щелкнуть по ней мышью. Подробные сведения о всех операторах и приемах работы в среде Visual Basic см. в разделе 4.

Оператор ***InputBox*** служит для ввода данных с клавиатуры в режиме диалога с пользователем. Простейший формат оператора для ввода переменных имеет следующий вид:

<имя переменной>Val(InputBox("текстовое сообщение")

Здесь Val – функция для перевода символьного выражения в числовое. При выполнении оператора *InputBox* на экран выводится окно диалога, в котором необходимо ввести требуемую числовую величину; <текстовое сообщение> - сообщение о том, что требуется ввести в строку ввода. При вводе символьных переменных функция Val не используется.

Пример 3.1. Использование оператораInputBox:

```
NG= Val(InputBox("Введите Ваш год рождения"))
C= InputBox("Введите фамилию автора книги")
```

Оператор вывода данных

Для вывода данных используются операторы ***Print***.

Оператор Print выводит данные на форму.

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.

В качестве выражений могут использоваться константы, переменные, функции, выражения. При выводе на форму числовых данных они должны переводиться в переменные символьного типа с помощью функции Str.

При отладке программы для вывода данных можно использовать метод Print объекта Debug:

Debug.Print <список переменных>

Переменные в списке разделяются запятыми

Пример 3.2. Использование оператора Print

```
Print "X=";Str(x)  ' выводится значение x
Print "X=";Str(x),"Y=";Str(y) ' выводятся значения x и y в соседних зонах
Print "X=" Str(x), "Y=";Str(Sin(x)) ' выводится значение x, вычисляется и
                                  ' выводится значение функции Sin(x)
Print "Автор"; S  ' выводится переменная символьного типа.
Debug.Print x, y  ' данные выводятся в окно непосредственного наблюдения
```

Примеры линейной программы

Пример 3.3. 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. 3.5).

Запись алгоритма решения задачи на алгоритмическом языке

```
алг обмен_данными
    дано A, B
    надо поменять значения A и B
нач
  T := A
  A := B
  B:= T
  рез A, B
кон
```

3. Запись алгоритма на языке программирования (Visual Basic 6.0):

```
        a= Val(InputBox( "Введите значение A "))
        b= Val(InputBox( "Введите значение B "))
    T=a
    a=b
    b=T
    Print "a=";a, "b=",b
```

4. Для отладки программы достаточно ввести значения a и b и проконтролировать результат.

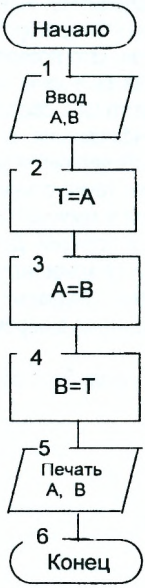


Рис.3.5 Схема алгоритма

Пример 3.4. 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:

$$p = 1/2(a + b + c) \qquad (3.2)$$

$$S = \sqrt{p(p-a)(p-b)(p-c)} \qquad (3.3)$$

Для оформления отчетной документации необходимо описать все переменные, используемые в программе по форме, представленной на (рис. 3.6). 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. 3.5.

3. Запись алгоритма на языке программирования:

```
Private Sub Form_Click()
Rem Вычисление площади треугольника
a= Val(InputBox( "Введите значение A "))
    b= Val(InputBox( "Введите значение B "))
c= Val(InputBox( "Введите значение C "))
p=(a+b+c)/2
S=Sqr(p*(p-a)*(p-b)*(p-c))
Print "S=";S
            End Sub
```

Чтобы сделать программу более удобной для чтения и понимания алгоритма, в нее рекомендуется вводить комментарии. Комментарии вводятся с помощью оператора ***Rem*** или символа апостроф (').

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= 0,5.

Пример 3.5. Вычисление первой и второй производных.

Если функция задана в виде таблицы, то производная от функции в заданной точке может быть вычислена численными методами как предел отношения конечных разностей $\Delta y=y_1-y_0$ и Δx: $y'= \Delta y/\Delta x$., где Δx – шаг, разность между соседними значениями аргумента, или, взяв приращения слева и справа от точки x, получают центральную разность:

$$f'(x) = \lim_{\Delta x \to \pm 0} \frac{(f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + 0(\Delta x) \qquad (3.4)$$

Отсюда вытекает способ численного дифференцирования. Если заменить предел Δx его конечным значением h, то получим приближенные формулы для вычисления первой и второй производных

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	
S	S	вещественная	площадь треугольника	

Рис. 3.6. Форма для описания переменных задачи

$$f'(x) \approx \frac{f(x+h) - (f(x-h)}{2h} \qquad (3.5)$$

$$f''(x) \approx \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} \qquad (3.6)$$

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.* Поэтому стремятся выбрать оптимальную величину шага, при которой ошибка усечения – Δ (ошибка, вызванная отбрасыванием правых членов разложения) и ошибка округления - ε будут примерно равны. Для формулы (3.4) оптимальный шаг определяется из выражения

$$h = \sqrt[3]{\frac{3\varepsilon}{M_3}} \qquad \text{или} \qquad \Delta = \frac{1}{6h}\left|\Delta^3 y\right| \approx \frac{1}{2}\frac{\varepsilon}{h}, \qquad (3.7)$$

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.

Простейшим способом выбора шага является выполнение неравенства $|f(x+h)-f(x)|>\xi$,, где ξ - некоторое малое число. При вычислении производной это исключает вычитание близких по значению чисел, которое обычно приводит к увеличению погрешности вычисления.

Пример 3.6. ЭВМ выводит результат с 8 знаками после запятой, при этом семь знаков точные, то есть ошибка округления - ▯ не превышает половины восьмого знака после запятой. Требуется определить значение шага при вычислении производной первого порядка, чтобы ошибка усечения не превышала 0,0001 (т. е. Δ<= 0,0001).

Решение. Абсолютная погрешность вычисления функции равна $0{,}5*10^{-7}$.

из (3.7) 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 (3.6) величина шага определяется из следующего соотношения

$$\Delta = \frac{1}{12\,h^2}\left|\Delta^4 y\right| \approx 4\,\frac{\varepsilon}{h^2}. \qquad (3.8)$$

В условиях примера 3.7. 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 и частного от деления двух чисел не превышает наибольшей из абсолютных погрешности сомножителей (делимого и делителя).*

Пример 3.7. Вычисление первой и второй производной в заданной точке.

```
REM Вычисление первой и второй производных по табличным данным
x= Val(InputBox( "Введите значение X "))   ' значение аргумента
y= Val(InputBox( "Введите значение Y "))   'значение функции в точке x
y1= Val(InputBox( "Введите значение Y1 ")) ' значение функции в точке слева от x
   y2= Val(InputBox( "Введите значение Y2 ")) ' значение функции в точке справа от x
   h= Val(InputBox( "Введите значение шага "))
    p1=(y2-y1)/(2*h)
    p2= (y2+y1-2*y)/h^2
    Print "Первая производная =";p1
    Print "Вторая производная =";p2
```

3.3.2. РАЗВЕТВЛЯЮЩИЕСЯ ПРОГРАММЫ

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 (две альтернативы), при сложном выборе – несколько. При наличии двух альтернатив задача выбора формулируется следующим образом:

Если <условие> *То* <действие первое> [, *иначе* <действие второе>]

Если условие выполняется, то выполняется действие первое, иначе выполняется действие второе. Здесь *если, то* и *иначе* - ключевые слова. Если число альтернатив больше двух, 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

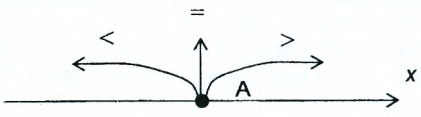


Рис. 3.7. Полная группа событий для точки на числовой оси

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*. В этом случае случайная величина *x* может принять по отношению к точке *A* одно из трех значений: меньше A, больше A или равно A. Это полная группа событий. Графическое представление этих условий приведено на рис. 3.7.

Условия записываются с помощью условных выражений. В условных выражениях в качестве операндов используются переменные и выражения, операнды соединяются знаками отношений: =, >, <, >=, <=, <>, операторами AND, OR, NOT.

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

Если группа событий полная, то для решения число проверок должно быть на единицу меньше, чем число исходов.

Пример 3.8. Написать программу для проверки условий согласно рис. 3.8.

Решение.

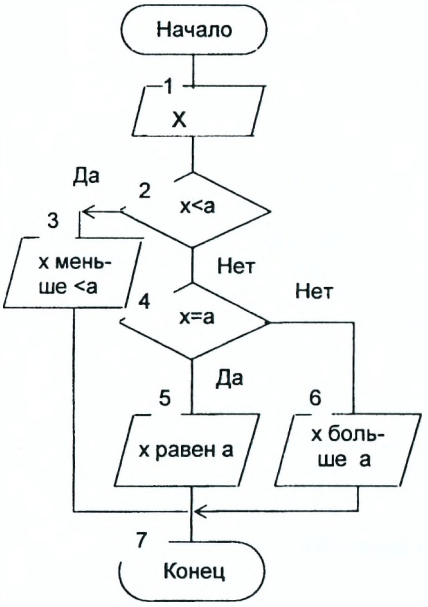


Рис. 3.8. Схема алгоритма к примеру 3.8

```
алг Проверка_значения X
  дано  x
    нач
  если   x<a
         то печать "X меньше a"
    иначе если x=a
     то  печать "X равен a"
  иначе
      печать "X больше a"
    все
кон
```

Алгоритм программы приведен на рис. 3.8.

Пример 3.9. 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
      иначе если i = 2
         то Текст1 = "Коня потеряешь": Вывод Текст1
      иначе если i = 3
         то Текст1 =  "Женату быть": Вывод Текст1
      иначе  Текст2 = "Нет выбора": Вывод Текст2
      все
кон
```

Схема алгоритма приведена на рис. 3.9.

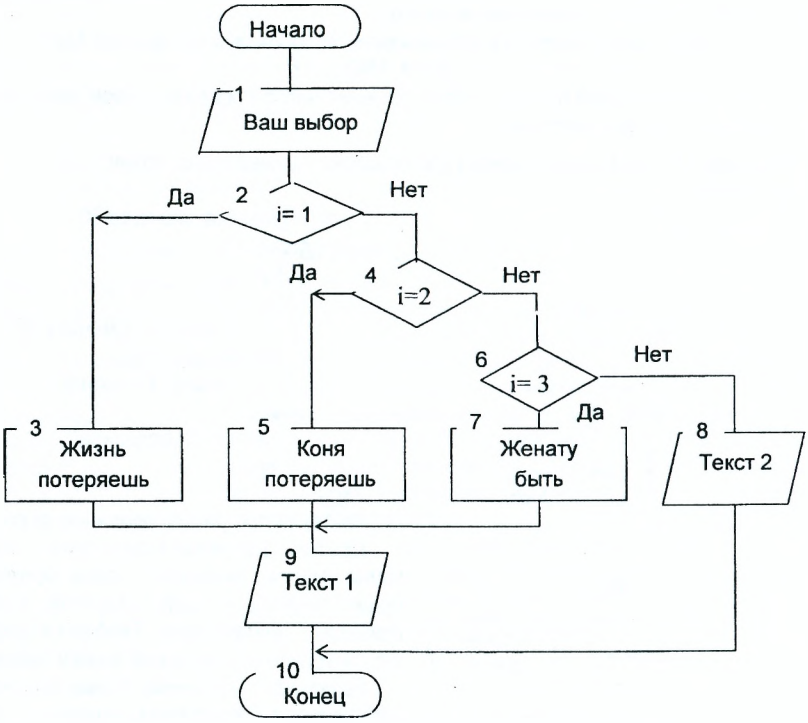


Рис. 3.9. Схема алгоритма к задаче 3.9

Пример 3.10. Вычислить значения функции

$$Y = \begin{cases} x^2-5, & \text{если } x<-2 \\ -x^2+5, & \text{если } -2 <= x <= 2 \\ x^3-5x-1, & \text{если } x>2 \end{cases}$$

Решение. В данной задаче группа событий полная, поэтому для вычисления значения Y при любых значениях аргумента достаточно выполнить две проверки.

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
  рез  y
кон
```

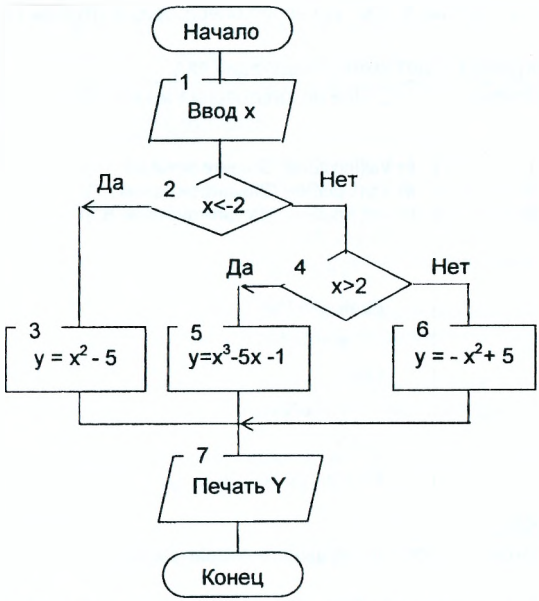


Рис. 3.10. Схема алгоритма к задаче 3.10

Алгоритм решения задачи 3.10 подобен алгоритму решения задачи 3.9 и приведен на рис 3.10.

Выбор решения в разветвляющихся программах осуществляется 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 опции 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, то выполняются операторы, указанные после опции Then, в ином случае выполняются операторы, следующие за опцией 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 опций Then и Else может быть указано несколько операторов, разделенных двоеточием. Однако число операторов ограничено длиной строки. Этих недостатков лишен многострочный оператор IF.

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.

Для условий задачи 3.10 программа будет иметь следующий вид:

А. При использовании однострочного оператора If	Б. При использовании многострочного оператора If
x= Val(InputBox("Введите значение X ")) a= Val(InputBox("Введите значение A ")) b= Val(InputBox("Введите значение B ")) If x<a Then y = x^2 –5: Goto m1 If x>b Then y=x^3-5*x-1: Goto m1 y = -x ^2+5 m1: Print "x="; x, "y="; y	x= Val(InputBox("Введите значение X ")) a= Val(InputBox("Введите значение A ")) b= Val(InputBox("Введите значение B ")) If x<a Then y = x^2 –5 ElseIf x>b Then y=x^3-5*x-1 Else y = -x ^2+5 End If Print "x="; x, "y="; y

Оператор выбора SELECT CASE

Оператор Select Case очень похож по структуре и выполняемым задачам на многострочный оператор If/Then/Else.

Оператор Select Case является оператором выбора. Он обеспечивает переход на ту или иную подпрограмму в зависимости от результатов вычислений. Так же как и многострочный If, оператор Select Case позволяет сформировать хорошо читаемую программу. В отличие от оператора If Оператор Select Case имеет большой набор опций для управления выбором. Очень удобен для обработки пунктов меню.

Синтаксис оператора Select Case

```
Select Case <переключающее выражение>
   Case 1  : REM 1 - условие выборки
       <Блок операторов 1>
   Case 2
       <Блок операторов 2>
       ...
   Case Else
    <Блок операторов n+1>
End Select
```

В качестве переключающего выражения может использоваться переменная числового или строкового типа или арифметическое выражение.

В качестве значения для блока Case могут использоваться следующие формы:

- число, или список значений, например: Case 1, 3 , 5, 8;
- выражение [, выражение]. *Выражение* – любое числовое или строковое выражение, которое должно совпадать с типом переключающего выражения;
- выражение TO выражение, например: Case 1 TO 10. Данное выражение означает, что значением может быть любое число из указанного диапазона;
- IS выражение. Ключевое слово IS представляет собой условие, например (Case IS <5).

Блок Case Else выполняется тогда, когда не выполнено ни одно из предыдущих условий.

Пример 3.11. Использование оператора Select Case.

Организовать вычисление определенного интеграла тремя методами: методом средних прямоугольников, методом трапеций, методом Симпсона.

```
Private Sub Form_Click()
    Cls  ' Очистка формы
    Print "Укажите метод вычисления интеграла: 1 - метод средних "
    Print "прямоугольников, 2 - метод трапеций, 3 - метод Симпсона"
    metodint = Val(InputBox("Укажите метод вычисления интеграла"))
    Print
    Select Case metodint    ' metodint - переключающее выражение
      Case 1
      Print "Вычисляется интеграл методом средних прямоугольников"
      '<программа вычисления интеграла методом средних прямоугольников>
      Case 2
          Print "Вычисляется интеграл методом трапеций"
        '  <программа вычисления интеграла методом трапеций>
      Case 3
          Print "Вычисляется интеграл методом трапеций"
        '  <программа вычисления интеграла методом Симпсона>
      Case Else
        Print "Неправильный ввод. Введите правильно номер варианта"
        Beep  ' Выдача звукового сигнала
        End Select
    End Sub
```

3.3.3. 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 классификация приведены в разделе 3.2.

Таблица 3.3 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

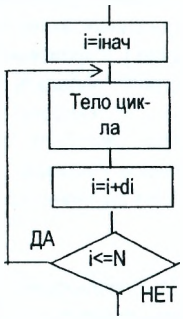


Рис. 3.11. Цикл с постусловием

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

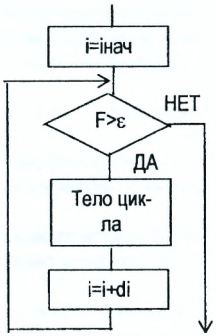


Рис. 3.12. Цикл с предусловием

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. 3.3.

Оператор FOR/NEXT

Оператор For служит для организации циклов с заданным числом повторений. Цикл относится к циклам с постусловием. Синтаксис оператора:

```
For i=Iнач To Iкон Step di
    <тело цикла>
Next i
```

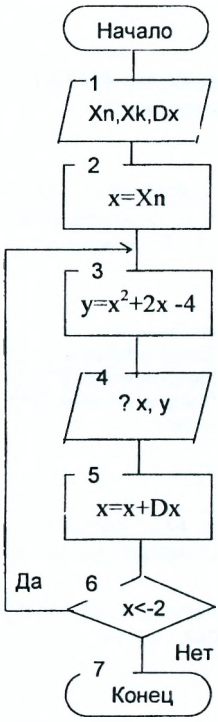


Рис. 3.13. Схема алгоритма к задаче 3.12

В данном формате Інач – начальное значение переменной цикла, Ікон – конечное значение переменной цикла, а di – шаг приращения значения переменной цикла.

Операторы For и Next образуют операторные скобки, между которыми заключено тело цикла.

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. Для вычисления значений тригонометрических функций, интегралов широко использовались таблицы, рассчитанные в научных центрах и представленные в виде таблиц, например, таблицы Брадиса. 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 значение 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 3.12. Составить таблицу значений функции $y= x^2+2x -4$ для *x*, изменяющегося в интервале 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. 3.13):

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.

К примеру 3.12

```
алг   Табулирование_функции
   дано  Xn, Xk, Dx
   нц   для x от Xn до Xk с шагом Dx
          y = x^2+2x -4
          рез x, y
   кц
кон
```

```
REM Табулирование функций
Xn= Val(InputBox( "Введите значение Xn "))
Xk= Val(InputBox( "Введите значение Xk "))
Dx= Val(InputBox( "Введите значение Dx "))
For x=Xn To Xk Step Dx
     y=x^2+2*x-4
     Print x, y
Next x
```

Пример 3.13. Напечатать таблицу перевода температуры из градусов по шкале Цельсия (C) в градусы по шкале Фаренгейта (F) для значений температуры от 15°C до 30°C с шагом 1°C. 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:

```
Private Sub Form_Click()
Tn=Val(InputBox( "Введите начальное значение температуры ° C"))
Tk =Val(InputBox( "Введите Конечное значение температуры ° C"))
Dt=Val(InputBox( "Укажите шаг изменения температуры ° C "))
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); F
Next i
End Sub
```

T°C	T°F
10	50
15	59
20	68
25	77
30	86

Вычисление сумм и произведений ряда чисел

Пусть требуется вычислить сумму и произведение натурального ряда чисел от 1 до N, то есть вычислить выражения

$$S = \sum_{i=1}^{N} i \ , \ i \in \overline{1,N} \tag{3.9}$$

$$P = \prod_{i=1}^{N} i, \ \ i \in \overline{1,N} \tag{3.10}$$

Для вычисления суммы чисел переменной S до входа в цикл присваивается начальное значение нуль S=0, а для вычисления 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.

К примеру 3.14	
алг Вычисление суммы *дано* Xn, N *нач* S=0 P=1 *нц* *для* i *от* Xn *до* N *с шагом* Dx Dx S=S+i P=P*i *кц* *рез* S, P *кон*	REM Вычисление суммы REM и произведения чисел Xn= Val(InputBox("Введите значение Xn ")) N= Val(InputBox("Введите значение N ")) Di= Val(InputBox("Введите значение Di ")) S=0: P=1 FOR i=Xn TO N STEP Di S=S+i P=P*i NEXT i PRINT "S=";S, "P=";P

Пример 3.14. Вычислить сумму и произведение чисел натурального ряда от 1 до N.
Решение. Введем обозначение Xn – начальное значение аргумента, N – число членов ряда, Dx – шаг, S – сумма чисел, P – произведение чисел.

Схема алгоритма приведена на рис. 3.14. Данный алгоритм не отвечает свойству массовости, так как не позволяет 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 вычислении 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, 6 MOD 2, - результат 0.

Пример 3.15. Вычисление суммы и произведения чисел

```
Private Sub Form_Click()
Rem Вычисление суммы  четных чисел
Cls ' Очистка формы
Print "Для вычисления суммы  введите число 1"
Print "Для вычисления произведения  введите число 2"
Print "Для вычисления суммы и произведения введите  число 3"
k=Val(InputBox("выберите вариант вычисления"))
Cls ' Очистка формы
Print "Для вычисления выражения для четных чисел  введите число 1"
Print "Для вычисления выражения для нечетных чисел введите число 2"
Print "Для вычисления выражения для любых чисел введите  число 3"
k1=Val(InputBox("Укажите вариант вычисления выражения"))
Cls
Xn= Val(InputBox( "Введите значение начального члена ряда "))
N= Val(InputBox( "Введите значение конечного члена ряда "))
```

```
Di= Val(InputBox( "Введите значение шага "))
S=0: P=1
For i = Xn To Xk Step Di
   Select Case k
     Case 1
        If i Mod 2 = 0 And k1 = 1 Then S = S + i
        If i Mod 2 > 0 And k1 = 2 Then S = S + i
        If k1 = 3 Then S = S + i
        'Print "S="; S
     Case 2
        If i > 0 Then
          If i Mod 2 = 0 And k1 = 1 Then P = P * i
          If i Mod 2 > 0 And k1 = 2 Then P = P * i
          If k1 = 3 Then P = P * i
          'Print "P="; P
        End If
     Case 3
        If i > 0 Then
     If i Mod 2 = 0 And k1 = 1 Then S = S + i: P = P * i
     If i Mod 2 > 0 And k1 = 2 Then S = S + i: P = P * i
     If k1 = 3 Then S = S + i: P = P * i
        'Print "S="; S, "P="; P
        End If
   End Select
Next i
If k = 1 Then Print "S="; S
If k = 2 Then Print "P="; P
If k = 3 Then Print "S="; S, "P="; P
End Sub
```

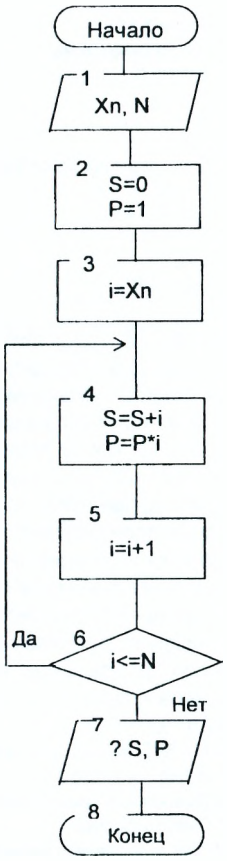


Рис. 3.14. Схема алгоритма к задаче 3.14

Оператор WHILE/WEND

Оператор WHILE служит для организации циклов с параметром (бесконечных циклов или циклы типа "ПОКА"), относится к циклам с предусловием. Условием окончания цикла является достижение функцией, вычисляемой в теле цикла, некоторого наперед заданного значения. Синтаксис 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 величины ε, то программа не войдет в цикл. Схема алгоритма приведена в табл. 3.3. рис. 3.12.

Оператор DO

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 опция WHILE, то тело цикла выполняется в том случае, если условие истинно. Когда используется опция 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 3.16. Составить таблицу значений функции e^x для заданного значения x с точностью 0,0001.

Решение. Функцию e^x, можно представить в виде ряда

$$e^x = 1 + \frac{x^1}{1} + \frac{x^2}{1*2} + \frac{x^3}{1*2*3} \ldots + \frac{x^i}{i!} + \ldots$$

и

$$e^x = 1 + \sum_1^{\infty} \frac{x^i}{i!} \qquad (3.11)$$

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.3.15).

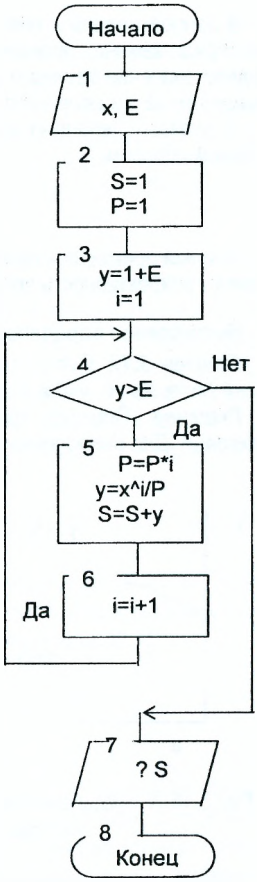


Рис. 3.15. Схема алгоритма к примеру 3.16

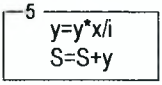
Первый блок вводит исходные данные: x - значение аргумента и E — точность вычисления значения текущего члена ряда. Второй и третий блоки осуществляют начальное присвоение переменной цикла, S – начальное значение суммы, P – вспомогательная переменная для вычисления факториала. Начальное значение суммы равно единице, так как в выражении 3.9 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:

$$e^x = 1 + \sum_1^{\infty} \frac{\prod_1^i x}{\prod_1^i i} = 1 + \sum_1^{\infty} \prod_1^i \frac{x}{i}, \qquad (3.12)$$

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. 3.15) будет выглядеть следующим образом:



Данный пример показывает, как от правильного построения математической модели зависит эффективность алгоритма.

Вычисление определенного интеграла

Пример 3.17. Пусть требуется 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".

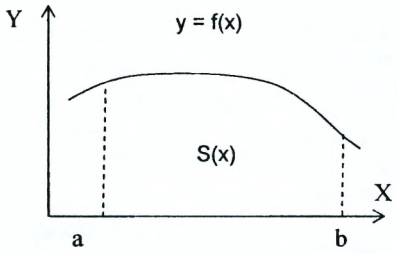


Рис.3.16. К определению определенного интеграла

Пусть функция *y = f (x)* неотрицательна и непрерывна на отрезке *[a; b]*. Тогда площадь *S (x)* криволинейной трапеции представляет собой функцию от *x* (рис. 3.16).

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. 3.16):

$$S = \int_a^b f(x)dx \qquad (3.13)$$

По определению $S = \int_a^b f(x)dx = F(b) - F(a)$,

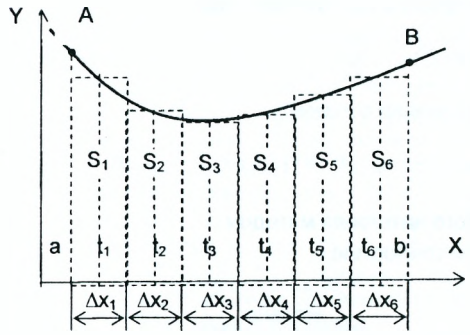


Рис. 3.17. Вычисление определенного интеграла методом средних прямоугольников

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 уда-ется 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. 3.17), 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

$$S_n = S_1 + S_2 + \ldots + S_{n-1} + S_n + R = \sum_{i=1}^{n} f(t_i)\Delta x + R \qquad (3.14)$$

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 $[a; b]$:

$$\int_a^b f(x)dx = \lim \sum_{i=1}^{n} f(t_i)x_i$$

$$\max \Delta x_i \to 0$$

Простейший метод численного интегрирования — *метод средних прямоугольников*. В нем используется замена интеграла интегральной функцией (3.14). В этом случае все отрезки Δx_i равны между собой, а в качестве $f(t_i)$ принимается значение функции в середине отрезка. Пусть h — длина элементарного отрезка, $h = (b-a) / n$; $x_{i-1/2}$ — значение аргумента в середине i-го отрезка, $x_{i-1/2} = a + (2i - 1)(b - a)/(2n)$, тогда можно записать:

$$\int_a^b f(x)dx = \sum_{i=1}^{n} h\, f(x_{i-1/2}) = \sum_{i=1}^{n} hf(a + hi - h/2) = \sum_{i=1}^{n} h_i f(a + (2i-1)h/2), \qquad (3.15)$$

Схема алгоритма приведена на рис. 3.15.

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:*

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n} h_i (f(x_i) + f(x_{i-1})) + R \qquad (3.16)$$

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 (3.15) после преобразования принимает вид:

$$S = h\left(\frac{y_0 + y_n}{2}\right) + \sum_{i}^{n-1} y_i , \qquad (3.17)$$

а формула Симпсона записывается следующим образом:

$$S = \frac{h}{3}(y_0 + y_{2m} + 2(y_2 + y_4 + \ldots + y_{2m-2}) + 4(y_1 + y_3 + \ldots + y_{2m-1})) \qquad (3.18)$$

Программа вычисления определенного интеграла методом средних прямоугольников с заданной точностью

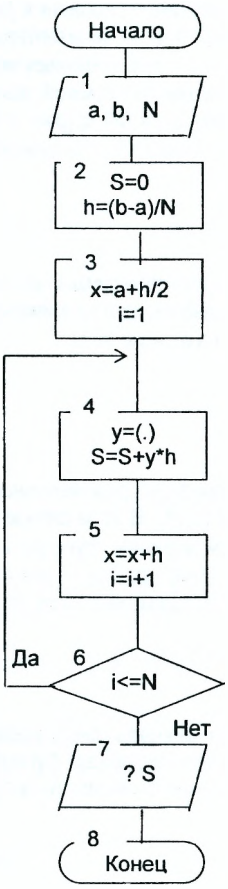


Рис. 3.18. Схема алгоритма к примеру 3.17

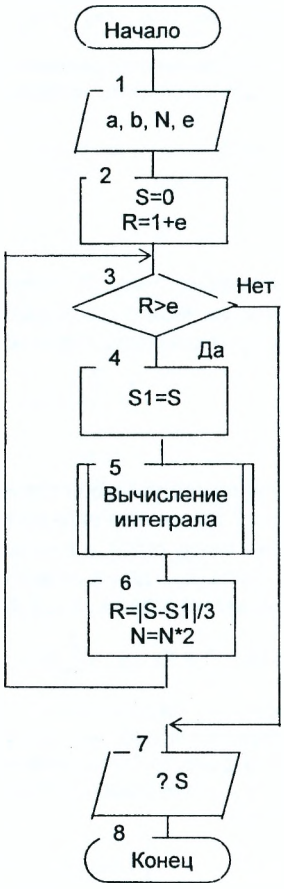


Рис. 3.19. Вычисление определенного интеграла

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 S при заданном числе отрезков разбиения N и результат запоминается в переменной S1. 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. Блок 5 на рис. 3.19 – предопределенный процесс (см. рис. 3.18).

```
        REM Вычисление определенного интеграла
        REM с заданной точностью
a= Val(InputBox( "Введите нижнюю границу отрезка интегрирования"))
b= Val(InputBox( " Введите верхнюю границу отрезка интегрирования "))
N= Val(InputBox( "Укажите число отрезков разбиения"))
e= Val(InputBox( "Укажите требуемую точность "))
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
```

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):

$$\Delta \approx \frac{1}{3} | I_n - I_{2n} | \text{ - для формулы трапеции,} \qquad (3.19)$$

$$\Delta \approx \frac{1}{15} | I_n - I_{2n} | \text{ - для формулы Симпсона.} \qquad (3.20)$$

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.

Корнем уравнения $f(x)=0$, где $f(x)$ - функция непрерывная и дифференцируемая на отрезке $[a,b]$ и в граничных точках, называется всякое число с, принадлежащее отрезку $[a,b]$, такое, что $f(c)=0$.

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.*

Отделение корней уравнений 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

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.

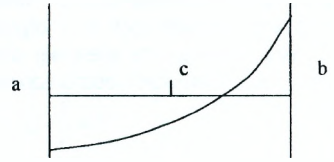
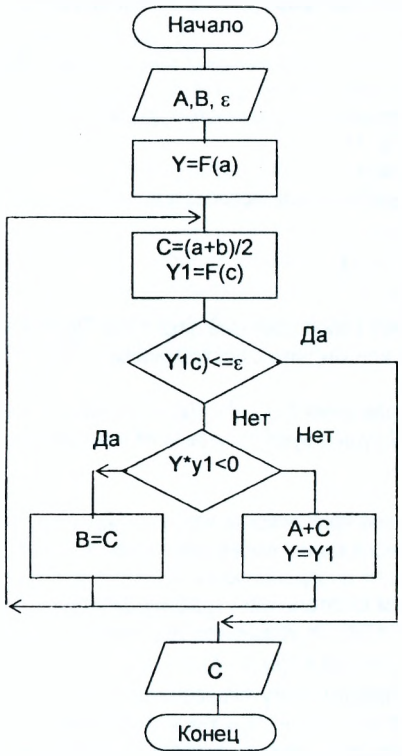


Рис. 3.20. К методу деления отрезка пополам

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.



3.21. 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 (рис. 3.20). 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 $c_1, c_2, c_3, \ldots, c_n$ будет сходиться к самому значению корня *c*. Алгоритм уточнения значения корня на отрезке отделения методом деления отрезка пополам приведен на рис. 3.21.

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:

$$x = \varphi(x), \qquad (3.21)$$

полученное путем эквивалентных преобразований из исходного уравнения *f(x) = 0*. В результате решения этого уравнения получаем ряд приближений корня:

$$x_{k+1}=\varphi(x_k), \qquad k=0,1,2,... \qquad (3.22)$$

Процесс уточнения корня прекращается, когда выполняется условие

$$|x_{k+1} - (x_k)| < \varepsilon \quad \text{или} \quad f(x_{k+1}) < \varepsilon \qquad (3.23)$$

Время поиска зависит от выбранного начального приближения x_0 и шага Δx.

Выражение вида (3.21) называется рекуррентной формулой. В итерационных алгоритмах рекуррентная формула имеет вид

$$x_{i+1} = \varphi(x_i). \qquad (3.24)$$

Пример 3.18. 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 (3.21)

$$x = \ln(x+1{,}1)/\cos(x) \qquad (3.25)$$

Представим выражение (3.25) в виде, удобном для использования в итерационном цикле:

$$x_i = \frac{\ln(x_{i-1} + 1{,}1)}{\cos(x_{i-1})} \qquad (3.26)$$

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. 3.22). Проведем касательную к графику функции в точке *B (b, f (b))*. Ее уравнение будет иметь вид:

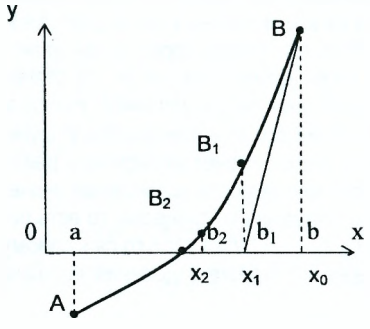


Рис. 3.22. Уточнение корня на отрезке методом касательной

$$y = f(b) + f'(b)(x - b). \qquad (3.27)$$

Найдем точку пересечения касательной с осью *x* (точка b_1). Так как в этой точке $y=0$, то, решая уравнение относительно *x*, получим:

$$x = b - \frac{f(b)}{f'(b)}. \qquad (3.28)$$

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 $x - x_2$.

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_2)}. \qquad (3.29)$$

Это второе приближение уравнения.

Для произвольной точки i получим

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \qquad (3.30)$$

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

$$x_0 - x_1 = bb_1 = Bb/tg(\alpha). \qquad (3.31)$$

На каждом шаге значение Δx_i меняется.

Таким образом, формула (3.30) дает последовательность приближений x_i корня. Процесс вычислений заканчивается, когда $f(x_{i+1})$ становится меньше некоторой наперед заданной величины ε, точности поиска корня.

Недостатком метода касательной является необходимость нахождения производных.

Приближенный поиск глобальных 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. 3.23 а) или внутри отрезка. Функция может быть выпуклая (выпуклая вверх) (рис. 3.23 б) или вогнутая (выпуклая вниз) (рис.3.23 в). Имеются простые способы анализа функций на отрезках, основанные на определении знака первой и второй производной.

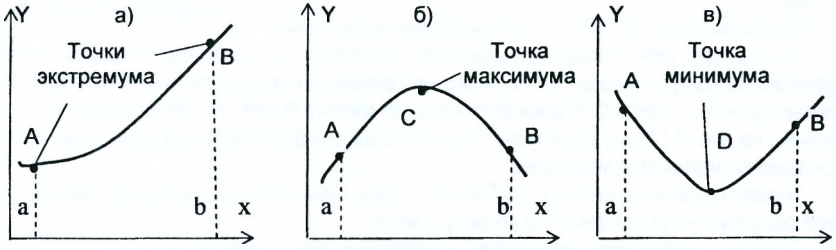


Рис. 3.23. К определению точек экстремума

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.

Приближенный поиск экстремума функции внутри отрезка может быть выполнен с помощью табулирования функции. 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.

Пример 3.19. Найти приближенное значение глобальных экстремумов функции y=F(x) на отрезке [A,B].

Решение. 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, например, Ymax=F(Xa), Ymin=F(Xa).

Алгоритм поиска глобального экстремума 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
```

```
REM Приближенный поиск
REM экстремума функции
A= Val(InputBox( "Укажите начало отрезка"))
B= Val(InputBox( " Укажите конец отрезка "))
Dx= Val(InputBox( "Укажите шаг табулирования"))
Ymax= (<выражение>): YMin=Ymax
FOR x=A TO B+Dx/2 STEP Dx
     y=(<выражение>)
     IF y>Ymax THEN Ymax=y
     IF y<Ymin THEN Ymin=y
NEXT x
PRINT "Ymax=";Ymax, "Ymin=";Ymin
```

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. 3.14., только в тело цикла включаются дополнительно проверки значений функции на минимум и максимум.

Точность поиска зависит от выбранного шага. Чем меньше шаг, тем точнее получаемое решение, но увеличивается время решения.

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 3.20. 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.

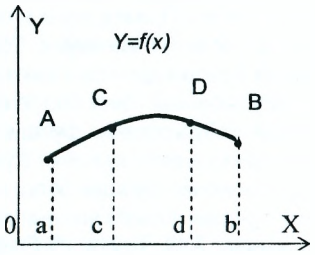


Рис. 3.24. Поиск экстремума функции методом "золотого сечения"

Метод "золотого сечения"

В отличии от метода итераций, метод "золотого сечения" не требует задания начального значения корня. Достаточно указать требуемую точность поиска. Анализируют не саму функцию, а ее производную. 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. 3.24) 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 производ-

ной 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.

Пример 3.21. Программа для поиска экстремума функции $Sin(x^2)+0,5$ методом "Золотого сечения"

```
Option Explicit
' Объявление типов переменных осуществляется в разделе Общие
' окна Программа
Dim x As Single, y As Single, a As Single, b As Single, e As Single
Dim k As Single, c As Single, d As Single, y1 As Single
'-------------------------------------------------------------------------
Private Function fnf(x)       ' функция пользователя
    fnf = SIN(x ^ 2) + .5
End Function
'-------------------------------------------------------------------------
Private Function fnf1(x)      ' производная от функции пользователя
    fnf1 = 2*x*COS(x ^ 2)
End Function
'-------------------------------------------------------------------------
Private Sub Form_Click()  ' Обработчик события Click формы
    REM Определение экстремума функции методом золотого сечения
    Cls
    a= Val(InputBox( "Укажите начало отрезка"))
    b= Val(InputBox( " Укажите конец отрезка "))
    e= Val(InputBox( "Укажите требуемую точность"))
    k = b     ' сохранение значения b
    m1:
    c = a + .382 * (b - a): d = a + .618 * (b - a)
    y = fnf1(a)                  ' fnf1 – производная от функция пользователя в точке a
    IF b - a <= e Then  x = (b + a) / 2: Print "экстремум: "; x, "y="; fnf(x): Exit Sub
                        ' печать результата и выход из программы
    b = c: y1 = fnf1(b)
    IF y * y1 < 0 Then Goto m1    ' проверяется отрезок [A,C]
    a = c: b = d: y = y1: y1 = fnf1(b)
    IF y * y1 < 0 Then Goto m1    ' проверяется отрезок [C, D]
    a = d: b = k: y = y1: y1 = fnf1(b)
    IF y * y1 < 0 Then Goto m1    ' проверяется отрезок [D,B]
End Sub
```

3.3.4. МАССИВЫ

Понятие об индексированных переменных

Массивом называется совокупность элементов, имеющих одно имя и отличающихся индексом.

Например двухмерный массив:

$$\begin{bmatrix} a_{11}\ a_{12}\ \dots\ a_{1j}\ \dots\ a_{1n} \\ \dots \\ a_{i1}\ a_{i2}\ \dots\ a_{ij}\ \dots\ a_{in} \\ \dots \\ a_{m1}\ a_{m2}\ \dots\ a_{mj}\ \dots\ a_{mn} \end{bmatrix} \qquad (3.32)$$

Здесь a_{ij} – элемент массива. Индекс i означает номер строки, j – номер столбца.

В данном разделе ограничимся лишь общими сведениями о массивах, необходимых для понимания сущности данного понятия, и способами их использования.

Массивы можно классифицировать по числу измерений и способу распределения памяти. По числу измерений массивы делятся на одномерные и многомерные, а по спо-

собу распределения памяти - на статические и динамические.

Число индексов в многомерных массивах ограничено, например, в системе Visual Basic оно не превышает шестидесяти. Верхнее значение индекса может быть равно 32767, а нижнее - 0. Индексы массивов целочисленные.

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.

Объявление массивов осуществляется оператором Dim. Размерность массива может задаваться константами или переменными. При объявлении массивов можно указывать и тип массива:

```
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.

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. Динамический массив объявляется дважды: первый раз в разделе общие формы без указания размерности, а второй раз в процедуре с указанием размерности оператором ReDim.:

```
Dim [Dinamic]  A () ' объявление без указания размерности
---------------------------------------------------------------
ReDim A(m,n) ' повторное объявление в процедуре с указанием размерности
```

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 3.22. Ввод и вывод одномерного массива.

```
10  Option Base 1 ' запись в разделе общие
11  Dim A()
---------------------------------------------------------------
20  Private Sub Command_Click() ' Обработчик события кнопки
30      Rem Ввод одномерного массива
40      N= Val(InputBox("Укажите размерность массива "))
50      ReDim A(N)
60      For  i = 1 TO N
70              A(i)= Val(InputBox("Введите " & Str( i) &  " элемент массива"))
80      Next i
```

Нумерация строк введена для удобства объяснения. По щелчку мыши на экран выводится окно диалога с запросом указания размерности массива, строка 40. Объявляется одномерный массив A размерностью N (строка 50). Затем организуется цикл с за-

данным числом повторений (строки 60 – 80). В строке 70 на экран выводится запрос на ввод значения элемента массива .

```
100       Rem Вывод одномерного массива
110       For  i = 1 TO N
120            Print  A(i),
130       Next i
140    End Sub
```

Алгоритм вывода массива на форму или печать подобен алгоритму ввода массива. В данном алгоритме необходимо обратить внимание на символ "запятая" 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 3.23. Ввод и вывод двухмерного массива

```
Option Base 1 ' запись в разделе общие
Dim A() As Single
-------------------------------------------------------------
Private Sub Command_Click() ' Обработчик события кнопки
      N= Val(InputBox("Укажите число строк "))
      M= Val(InputBox("Укажите число столбцов"))
        ReDim A(n, m)
        Rem Ввод данных в двухмерный массив
        For  i = 1 To n
           For  j = 1 To m
              A(i)= Val(InputBox("Введите a(" & Str( i) & _
                    "," & Str(j) & ") элемент массива"))
           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
End Sub
```

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.

3.3.5. ОПЕРАЦИИ С МАССИВАМИ

Массивы широко используются для хранения и обработки данных. При работе с файлами данных, вводе и выводе информации удобнее всего записывать данные предварительно в массив.

Умножение массива на число, вектор, массив

В последующих примерах операции объявления массивов, ввода и вывода данных из массивов опущены.

Пример 3.24. Умножить все элементы двухмерного массива на число С.

Решение. Для умножения массива на число необходимо умножить на это число каждый элемент массива.

```
For i = 1 To n
    For j = 1 To m
        A(i,j) = A(i, j)*C;
    Next j
    Print
Next i
```

Пример 3.25. Найти скалярное произведение векторов A и B.

Решение. При скалярном умножении вектора на вектор получается переменная, значение которой равно сумме произведений каждого элемента второго вектора на соответствующий элемент первого вектора: $C= \sum a_i * b_i$

```
C=0
For i = 1 To n
    C=C+A(i)*B(i)
Next i
```

Пример 3.26. Умножить двухмерный массив A на вектор B, результаты поместить в вектор C.

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: $A \cdot x = b$.

```
FOR i = 1 TO n
    FOR j = 1 TO m
        C(i)=C(i)+ A(i, j)*B(j);
    NEXT j
NEXT i
```

Пример 3.27. Умножить двухмерный массив A на массив B, результаты поместить в массив C.

Решение. При умножении массива A на массив B необходимо чтобы число столбцов в массиве A было равно числу строк в массиве B. При умножении массива A на массив B получается новый массив C, в котором число строк равно числу строк первого массива, а число столбцов - числу столбцов второго массива, а каждый элемент этого массива равен сумме произведений каждого элемента строки массива A на соответствующий элемент столбца B, т. е. $c_{ik}= \sum a_{ij} * b_{jk}$.

```
REM Умножение массива на массив
N= Val(InputBox("Укажите число строк в массиве A "))
M= Val(InputBox("Укажите число столбцов в массиве A"))
P= Val(InputBox("Укажите число столбцов в массиве B"))
ReDIM A(n, m) As Variant
ReDim B(m,p) As Variant
ReDim C(n,p) As Variant

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

Пример 3.28. Вычислить сумму элементов одномерного массива.

Решение.

```
S=0
FOR i = 1 TO n
    S=S+ A(i)
NEXT i
PRINT S
```

Задача 3.29. 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. настоящий раздел, пример 3.14).

```
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

Пример 3.30. Найти максимальный и минимальный элементы одномерного массива.

Решение. При поиске минимального и максимального значений элементов массива необходимо ввести вспомогательные переменные, например Amax и Amin, и присвоить им в качестве начального значения значение любого элемента массива, например, нулевого. Для запоминания индекса максимального или минимального элемента массива также необходимо ввести вспомогательные переменные:

```
Amax=A(0): Amin=A(0)
For i = 1 To n
    If A(i)>Amax  Then Amax=A(i): Imax=i
    If A(i)<Amin  Then Amin=A(i): Imin=i
Next i
Print "Imax"; Imax, "Amax =";Amax
Print "Imin"; Imin,  "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. Алгоритм обмена данными рассмотрен в примере 3.3.

Самая простая процедура сортировки – перебор.

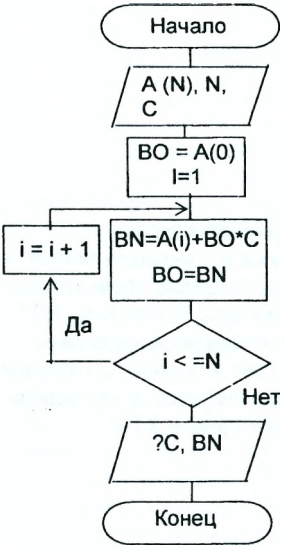


Рис. 3.25. Алгоритм вычисления многочлена

```
FOR i=1 To n - 1
        FOR j = i + 1 To n
         IF A(j) < A(i) THEN T=A(j): A(j)=A(i) :
           A(i)=T
    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.

Вычисление многочленов. Схема Горнера

Пример 3.31. 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 VB 6.0 – и дело сделано. Однако для повышения скорости вычисления целесообразно заменить операции умножения операциями сложения и вычитания. Особенно это актуально в управляющих ЭВМ, работающих в реальном масштабе времени.

Для вычисления многочлена используется *схема Горнера* (рис.3.25).

Пусть задан многочлен:

$$P(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n \qquad (3.34)$$

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:

$b_0 = a_0$

$b_1 = b_0 c + a_1$

$b_2 = b_1 c + a_2$

...

$b_n = b_{n-1} c + a_n$.

Отсюда $b_n = P(c)$.

Таким образом, вычисление значения многочлена $P(x)$ при $x = c$ сводится к повторению элементарных операций:

$$b_i = a_i + b_{n-1} c, \text{ где } i = 1,2, \ldots, n, \; b_0 = a_0.$$

Алгоритм вычисления многочлена $P(x)$ при $x = c$ по схеме Горнера приведен на рис. 3.25.

Пример вычисления многочлена по схеме Горнера:

В общем виде:						В числовом выражении				
	a_0	a_1	a_2	...	a_n		5	2	-15	-6
	+						+			
c		$b_0 c$	$b_1 c$	...	$b_{n-1} c$			12,5	36, 25	53,125
						2,5				
	b_0	b_1	b_2	...	$b_n = P(c)$		5	14,5	21,25	47,125

То есть P(2.5) = 47,125

A(N) – вектор коэффициентов многочлена, N – степень многочлена, C – значение аргумента, BO – коэффициент b_i, BN – текущее значение многочлена.

3.3.6.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 в точках, отличных от узлов интерполяции, 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

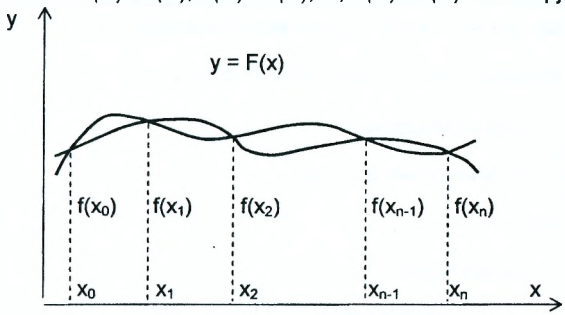


Рис. 3.26. Интерполирование

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 соответствует нахождение плавной кривой, проходящей через заданные точки $(x_i ; f(x_i))$ (рис. 3.26). 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.

Интерполяция функций методом Лагранжа

Интерполяционный многочлен Лагранжа имеет следующий вид:

$$L_n(x) = \sum_{i=0}^{n} \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}. \qquad (3.35)$$

При n=1 получаем формулу для линейной интерполяции:

$$L_1 = y_0 \frac{x-x_1}{x_0-x_1} + y_1 \frac{x-x_0}{x_1-x_0}. \qquad (3.36)$$

Для ускорения вычислений многочлена при различных значениях n применяют схему Эйткена, в которой не требуется явного построения многочлена:

$$y(x,x_0,x_1)=\frac{1}{x_1-x_0}\begin{vmatrix} y_0 & x_0-x \\ y_1 & x_1-x \end{vmatrix},$$

$$y(x,x_0,x_2)=\frac{1}{x_2-x_0}\begin{vmatrix} y_0 & x_0-x \\ y_2 & x_2-x \end{vmatrix}, \qquad (3.37)$$

$$y(x,x_0,x_1,x_2)=\frac{1}{x_2-x_1}\begin{vmatrix} y(x,x_0,x_1) & x_1-x \\ y(x,x_0,x_2) & x_2-x \end{vmatrix}.$$

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:
F1 – значение функции
Программа

```
Rem интерполяция полиномом Лагранжа по Эйткену
Dim X(), F()
-------------------------------------------------------------
Private Sub Form_Click()
  N=Val(InputBox( "Укажите число интервалов интерполяции"))
  ReDim X(N): ReDim F(N)
  For i=0 TO N
     X(i)=Val(InputBox( "Укажите значение X  в "  & Str(i) & " узле"))
     F(i)=Val(InputBox( "Укажите значение Y  в "  & Str(i) & " узле"))
  Next i
  A=Val(InputBox( "Укажите значение аргумента x "))
  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
End Sub
```

Интерполяционная формула Ньютона

Интерполяционная формула Ньютона имеет следующий вид:

$$P_{n-1}(x)=y_1+(x-x_1)f(x_1;x_2)+(x-x_1)(x-x_2)f(x_1;x_2;x_3)+(x-x_1)(x-x_2)\dots(x-x_{n-1})f(x_1;x_2;\dots;x_n)=\sum_{i=0}^{n-1}A_k\varphi_i; \qquad (3.38)$$

где $A0=y_1$, $A_k=f(x_1;x_2;\dots;x_{k+1})$ – 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.

Программа:

```
Dim X(), Y()
Rem подпрограмма интерполяции по Ньютону
N=Val(InputBox( "Укажите степень полинома"))
   ReDim X(N): ReDim Y(N)
For i=0 TO N
X(i)=Val(InputBox( "Укажите значение X  в "  & Str(i) & " узле"))
Y(i)=Val(InputBox( "Укажите значение Y  в "  & Str(i) & " узле"))
Next i
A=Val(InputBox( "Укажите значение аргумента x "))
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
```

3.3.7. 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 могут использоваться операторы Let и InputBox

При использовании оператора *Let* символьная переменная заключается в кавычки:

```
Let A$ = "Брестская крепость – герой"
```

При вводе символьных переменных с помощью оператора InputBox строковые переменные записываются без кавычек. Данные вводятся в строку ввода окна диалога.

```
S= InputBox("Введите фамилию первого летчика-космонавта")
```

Переменная S должна быть переменной символьного типа.

Вывод символьных переменных

Для вывода символьных переменных на экран могут использоваться операторы Print и метод Print объекта Debug.

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 "День победы! Как он был от нас далек..."
```

Функции для обработки символьных переменных

Visual Basic имеет значительное число функций для работы с символьными переменными. Рассмотрим некоторые из них.

Примечание 1. В данном разделе все строковые переменные будут обозначены символом C, а числовые переменные символом N.

Примечание 2. Ряд функций могут использоваться для обработки как строк символов, так и числовых данных. Поэтому они имеют два синтаксиса, например:

Ltrim(Строка любого типа)

Ltrim$(Строка символьного типа) As String

В первом случае функция может обрабатывать строки любого типа: числовые или строковые. Во втором случае в качестве аргумента должна быть строка символьного типа, и возвращает функция также строку символьного типа.

По назначению функции для работы с символьными переменными можно разделить на ряд групп.

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.

Функции для обработки строк: Len, Left$, Right$, Mid$, Ltrim$, Rtrim$, Instr; Lcase$, Ucase$, Instr.

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, начиная с позиции N1. Например:

```
C1 = Mid$("Шумит, гудит зеленый лес",8,5)
Print C1
```

Результат: гудит

Функция Mid$ выделяет из строки "Шумит, гудит зеленый лес" 5 символов с восьмой позиции - слово "гудит".

Функция Mid$ может использоваться также 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 С1, начиная с позиции N1. Если длина строки С1 больше N2, то лишние символы отбрасываются. Если длина строки С1 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
```

Результат: 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 возвращается 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
```

Результат: 8 - то есть начальная позиция слова "месяц" – восемь.

```
C = "светит месяц, светит ясный"
C1 = "светит"
n = Instr(C$, C1$)
        Print n
```

Результат: 1

Найдено первое вхождение слова "светит" в строку С. Для поиска других вхождений необходимо организовать цикл.

Функции генерирования строк символов: Space$, String$

Функция *Spase$(n)* генерирует строку из *n* пробелов.

Функция *String$* 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 имеют 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 алгоритмы обработки символьных переменных.

Пример 3.32.. Выделить i –й символ из текста.

Задача решается с помощью функции Mid$:

C1 = Mid$(C, i, 1)

Пример 3.33. Найти позицию, в которой располагается заданный символ или цепочка символов.

Задача решается с помощью функции Instr:

n = Instr(C1, C2)

Пример 3.34. Удалить символ или цепочку символов.

Решение:

- найти позицию, с которой заданный символ или цепочка символов входит в строку (функция Instr(C,C1));
- выделить часть строки слева от символа (функция Left$(C,N));
- выделить часть строки справа от символа (функция Right$(C,N));
- объединить полученные строки.

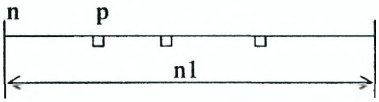
Пример 3.35. Вставить текст между i – м и i+1 – м символами.

- 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.

Пример 3.36. 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

```
Private Sub Form_Click()
    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$, n, n1 - (n - 1)): Exit Sub
        Print Mid$(c$, n, p - n)
        n = p + 1
        Goto m
    End IF
End sub
```

Пример 3.37. Сравнить символы или цепочки символов:
: "a"<"b"; "ab"="ab"; "ac">"ab"; "abc"<"abd", но "abf">"abd"
"abc"<"abcd", но "fbc">"abcd"

Пример 3.38. Выделить слово из текста.

Решение.

- вычислить длину слова (функция LEN);
- найти позицию вхождения слова в строку символов (функция INSTR);
- выделить слово (функция MID$).

Пример 3.39. Определить, является буква гласной или согласной.

- ввести строку текста;
- ввести строку символов, содержащую только гласные Cg$;
- выделить один символ из текста в позиции i (функция Mid$(C,i,1));
- проверить, принадлежит ли символ строке гласных Cg$ (функция Instr(C,C1)). Если принадлежит, значит символ – гласная.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Что такое табулирование функций? Как оно осуществляется?
2. Поясните порядок вычисления многочлена по схеме Горнера.
3. В чем различие между аналитическими и численными методами решения уравнений?
4. Что такое отделение корней уравнения?
5. Что понимается под уточнением корней уравнения?
6. Отделите корни уравнения $2x^3 + 5x^2 - 3=0$.
7. Отделите корни уравнения $3x^3 - 2x^2 + 5=0$.
8. Назовите методы уточнения корней уравнения на отрезке отделения.
9. В чем сущность метода дихотомии?
10. В чем отличие метода "золотого сечения" от метода дихотомии?
11. В чем сущность метода простых итераций?
12. Назовите условия наличия экстремума внутри отрезка.
13. Сформулируйте условия выпуклости и вогнутости функции.
14. Что такое интеграл? Дайте геометрическую интерпретацию определенного интеграла.
15. Поясните способы вычисления определенного интеграла.
16. Что такое интерполирование функций?
17. Что такое символьная переменная?
18. Приведите классификацию функций для обработки символьных переменных.
19. Что лежит в основе сравнения символьных переменных?
20. На чем основан принцип обработки символьных переменных?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с основами программирования. Основными элементами языка программирования являются команды, операторы, функции, выражения. Программа представляет собой последовательность операторов. Для реализации программы используется несколько базовых структур: линейные, выбор, вычисляемый переход, циклы.

Для ввода данных используются операторы Let, Input, Line Input Data, Read, Restore, функция Input$.

Для вывода данных используются операторы Print, Print Using, Lprint.

Процедуры выбора реализуются с помощью оператора If/Then/Else. Оператор If имеет две структуры: однострочный If и многострочный If. Многострочный If позволяет создавать хорошо структурированные, легко читаемые и поверяемые программы.

Циклы реализуются с помощью операторов For/Next, While/Wend, Do/Loop. Оператор For/Next создает циклы с постусловием, используется для организации циклов с заданным числом повторений. Оператор While/Wend служит для организации циклов с параметром (бесконечных циклов). Данный оператор позволяет создавать циклы с предусловием и реализует положительную логику, то есть тело цикла выполняется в том случае, когда условие выполняется. Оператор Do/Loop – универсальный оператор. Он позволяет создавать как циклы с предусловием, так и циклы с постусловием. При этом могут создаваться циклы как с положительной логикой, так и с отрицательной логикой.

Названные операторы позволяют создавать программы для решения самых разных задач: вычисления арифметических и алгебраических выражений, табулирования функций, исследования функций, решения задач математического анализа численными методами, работы с массивами.

Основная трудность в усвоении вопросов программирования лежит не в изучении операторов, а в знании методов математического анализа и умении составлять алгоритмы программ.

4. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ VISUAL BASIC 6.0

4.1. ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ ПРОГРАММИРОВАНИЯ VISUAL BASIC 6.0

Изменение технических возможностей вычислительной техники способствовало разработке и новых систем программирования, позволяющих инженеру не программисту самостоятельно разрабатывать пользовательские программы достаточно высокого качества для решения текущих задач. Одним из таких языков программирования высокого уровня является язык Visual Basic (VB).

К настоящему времени существует уже несколько версий языка программирования Visual Basic: VB1, VB3, VB4, VB5 и VB6, VB7.

Visual Basic 6.0 опирается на язык высокого уровня – Basic. Синтаксис и операторы этого языка по структуре близки к алгоритмическому языку программирования, изучаемому в школах, что позволяет гораздо легче усваивать его начинающим пользователям по сравнению с такими профессиональными языками программирования как Pascal, Delphi, C. Актуальность изучения языка VB обусловлена не только тем, что он достаточно прост в изучении и обладает большими возможностями по разработке прикладных программ, но также тем, что он используется для написания макросов во всех популярных приложениях Windows: Word Basic – для текстовых редакторов; Basic for Application – для электронных таблиц; Visual Basic – для баз данных. Возможность использования элементов ActiveX, распространяемых по сети Интернет, открывает перед пользователем практически неограниченные возможности по разработке приложений любой сложности.

Visual Basic сочетает в себе возможности транслятора интерпретирующего типа (интерпретатора) и транслятора компилирующего типа (компилятора).

Как интерпретатор VB позволяет запускать программы непосредственно из среды разработки. Интерпретатор принимает непосредственное участие в выполнении программы. Он разбирает каждую строку, проводит анализ ошибок и выдает сообщение пользователю при их обнаружении. После устранения некритических ошибок он продолжает работу. Большая часть времени процессора тратится именно на разбор и трансляцию каждой строки программы при каждом ее повторном исполнении. Поэтому программы с трансляторами-интерпретаторами работают значительно медленнее, чем скомпилированные программы. Но у трансляторов-интерпретаторов больше возможностей в организации диалогового режима отладки программ. При запуске программы из среды разработки, VB сразу же проверяет синтаксис программы и выдает сообщение об обнаруженной ошибке. За счет применения специальных технологий созданы благоприятные условия для поиска ошибок.

Как компилятор VB позволяет создавать EXE-файлы, правда не сразу при запуске из среды разработки, а с помощью команд меню. EXE-файлы выполняются под управлением операционной системы без участия транслятора. Процессы написания, отладки и выполнения программы разделены во времени.

Visual Basic 6.0 имеет три различные редакции, рассчитанные на различные группы пользователей и отличающиеся набором возможностей и комплектом поставляемой документации: VB для начинающих, VB для профессионалов и промышленное издание VB. При этом синтаксис языка VB остается неизменным и не зависит от издания. Промышленное издание представляет собой расширенное издание для профессионалов и предназначено для разработчиков корпоративных систем. Набор элементов управления и средств этой версии позволяет разрабатывать не только простейшие программы, но и достаточно сложные клиент-серверные приложения.

Visual Basic 6.0 предъявляет достаточно высокие требования к техническим характеристикам персонального компьютера. Для установки VB требуется не менее 10 Мбайт дисковой памяти, процессор не хуже 80486 или Pentium, 16 Мбайт памяти оперативного запоминающего устройства. Полная инсталляция самой мощной версии VB требует более 100 Мбайт дискового пространства.

Visual Basic работает в среде Windows (не ниже Windows 95) и позволяет создавать 32-х разрядные ***приложения***[5] – программы для работы в этой среде. При этом программы имеют похожий интерфейс и способы управления. В частности, VB позволяет добавлять к окнам поля ввода, меню, командные кнопки, переключатели, флажки, списки, линейки прокрутки, а также диалоговые окна для выбора файла или каталога. Программист может использовать сетку для обработки табличных данных, организовать взаимодействие с другими приложениями Windows и доступ к базам данных. Visual Basic, начиная с версии VB5, может поддерживать элементы ActiveX. ***ActiveX*** - компонент представляет собой технологию Microsoft для активизации работы с Internet и корпоративными Internet-сетями, причем данная технология может использоваться в обычных приложениях Windows для повышения продуктивности работы. Эта технология связана с вставкой и внедрением объектов. ActiveX компонент предоставляет в распоряжение пользователя набор компонентов многократного использования и возможности простой работы с этими компонентами. Однако рассмотрение этих богатых возможностей VB выходит за рамки предлагаемого пособия.

4.1.1. ЗАПУСК ПРОГРАММЫ

Запуск программ VB можно осуществить разными способами.

Если программа установлена на компьютере, то ее можно запустить через команду главного меню: ***Пуск , Программы , VB6*** либо через программу ***Проводник*** или ***Мой компьютер***.

При запуске программы через Проводник, Мой компьютер или меню кнопки Пуск на экране появляется окно запуска ***New Project*** (рис.4.1). В этом окне можно выбрать тип создаваемого приложения. Окно содержит три закладки:

New – начать с "нуля";

Existing – работать с существующим проектом;

Recent – работать с проектом, разработка которого проводилась недавно.

То же самое происходит при запуске новой программы из среды разработки, если выбрать команду ***File / New Project,*** однако в этом случае окно не имеет закладок.

Для начала работы с новым проектом следует выбрать закладку ***New*** и выбрать тип ***Standard.EXE***.

Можно выбрать также Мастера приложений – ***VB Application Wizard***. В процессе работы мастера создается почти готовое приложение с различными формами, соответствующей рабочей средой, меню, панелью инструментов и т.д., которое можно потом совершенствовать. Однако пользоваться этим мастером рекомендуется после приобретения определенного опыта работы с Visual Basic. В ином случае пользователь получит больше проблем по совершенствованию проекта, чем пользы. Поэтому начинающему пользователю рекомендуется начинать работу со стандартного окна Standard.exe.

---

[5] Программы, работающие под управлением Windows, принято называть приложениями.

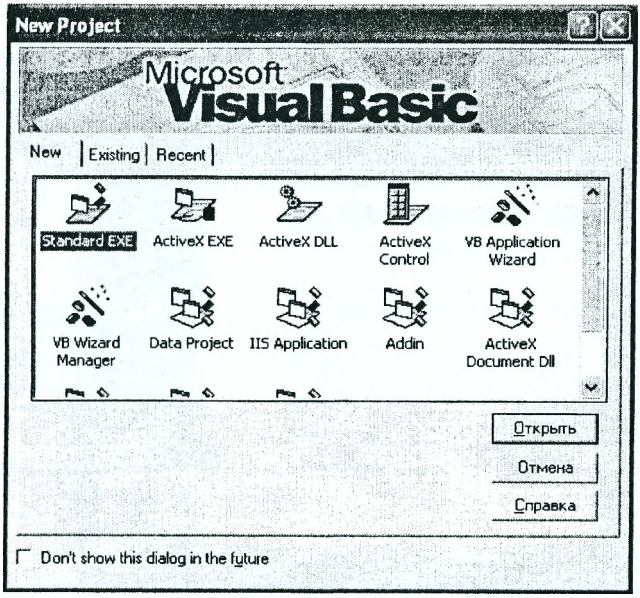


Рис. 4.1. Так выглядит диалоговое окно New Project

Выход из программы

Выход из VB осуществляется комбинацией клавиш ***Alt-F4*** или командой ***File/Exit.***

4.1.2. СРЕДА РАЗРАБОТКИ

Рабочее окно

После выбора типа проекта появляется рабочее окно программы (рис. 4.2). Рабочее окно представляет собой *интегрированную среду разработки* (IDE) – интерфейс самого продукта Visual Basic. Эта среда может настраиваться с помощью диалогового окна, вызываемого командой ***Tools/Options.*** К этому окну диалога мы будем обращаться по мере необходимости. Рабочее окно предлагает целый набор инструментальных средств, которые можно использовать при создании программ.

Большинство элементов рабочего окна, характерны для приложений Windows: строка заголовка (2), кнопка (1) – кнопка системного меню, кнопки (3, 4, 5) предназначены для свертывания, развертывания и закрытия окна, соответственно. После развертывания окна появляется новая кнопка для восстановления первоначальных размеров окна, меню (13), панели инструментов: стандартная панель инструментов -Toolbar (12) и панель элементов управления Toolbox (9).

Внутри окна программы открываются вторичные окна: окно конструктора форм (10) с макетом формы - ***Form1*** (11), окно проекта – ***Project1(Project1)*** (6); окно свойств – ***Properties*** (7); окно макета формы ***Form Layout*** (8).

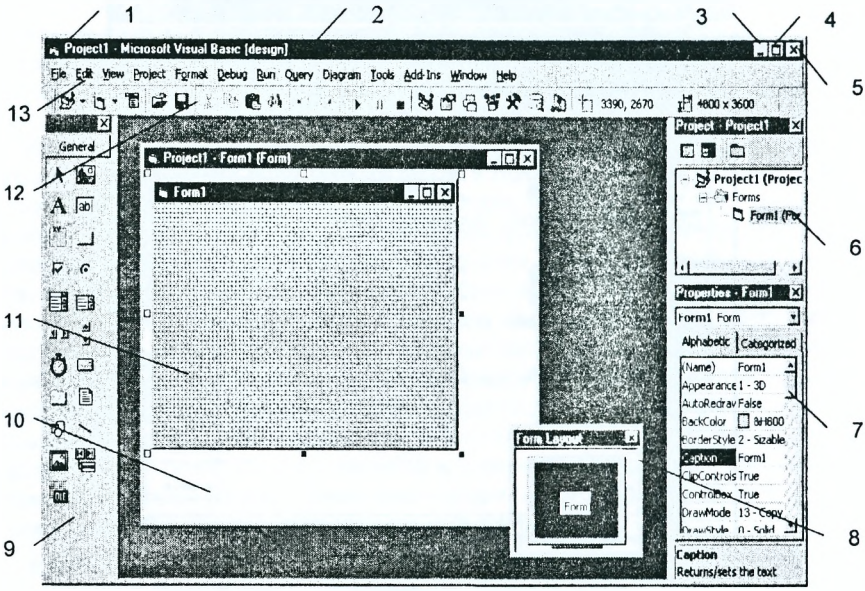


Рис. 4.2. Окно программы Visual Basic

1 – кнопка системного меню, 2 – заголовок, 3 – кнопка свертывания окна в пиктограмму, 4 – кнопка развертывания окна, 5 – кнопка закрытия окна, 6 – окно проекта, 7 – окно свойств, 8 – окно позиционирования формы, 9 – панель элементов управления, 10 – окно конструктора форм, 11 – макет формы, 12 – стандартная панель инструментов, 13 – меню

Меню

Строка Меню содержит список команд, предназначенных для управления разработкой проекта, пункты меню содержат несколько уровней вложения:

File (Файл) - содержит команды для работы с файлами создаваемых приложений, загрузки, сохранения, вывода на печать.

Edit (Правка) - содержит команды редактирования, предназначенные для создания исходного текста программы, включая средства поиска и замены.

View (Просмотр) - обеспечивает доступ к различным частям приложения и среды разработки VB.

Project (Проект) - предназначен для добавления новых объектов VB к разрабатываемым проектам, добавления или удаления элементов управления на панель элементов управления, настройки свойств проекта.

Format (Формат) – дает доступ к различным настройкам элементов управления, размещенных на создаваемых программистом формах.

Debug (Отладка) – содержит средства, предназначенные для отладки программ или поиска ошибок.

Run (Выполнение) – служит для запуска и остановки программ непосредственно из среды разработки.

Tools (Инструменты) – обеспечивает доступ к работе с процедурами и меню внутри приложения. Этот пункт меню имеет важную команду ***Options,*** которая открывает одноименную диалоговую панель с закладками ***Options***, где настраивается практически вся среда разработки Visual Basic.

Add-Ins (Модули) - дает доступ к инструментам, которые могут быть добавлены к окружению VB: мастера, ActiveX – элементы и другое. По умолчанию в него включается команда Visual Data Manager и Add-In Manager. Visual Data Manager – мастер для проектирования и заполнения баз данных. Add-In Manager – менеджер модулей служит для выбора других надстроек, включаемых в модули.

Diagram (Диаграмма) – содержит средства для оформления отчетов: форматирования текста, таблиц, диаграмм.

Window (Окно) – используется для работы с окнами в среде разработки.

Query – доступ к внешним базам данных.

Help – справочная система.

Выбор пунктов меню осуществляется мышью или с помощью клавиатуры. При управлении с помощью клавиатуры для входа в меню используется клавиша ***Alt***. Выбор пунктов меню осуществляется с помощью клавиш управления курсором или с использованием клавиатурных команд (комбинация клавиши Alt и символа подчеркнутого в командах меню: [***Alt- клавиша***]).

Панели инструментов

VB имеет четыре стандартные панели инструментов: *Standard* (стандартная) – стандартная, *Edit (правка)* - редактирования, *Debug* (отладка)– отладки и *Form Editor* (редактор форм) – редактор форм. Вывод панелей инструментов осуществляется следующим образом: выберите пункты меню ***View, Toolbars*** и щелкните в открывшемся меню третьего уровня мышью по наименованию нужной панели инструментов или выберите опцию ***Customize*** (общие) и установите флажки у тех панелей, которые нужно отобразить на рабочем столе.

Панели инструментов Edit, Form Editor, Debug открываются также при выборе пунктов меню ***Редактирование*** (Edit), ***Формат*** (Format) и ***Отладка*** (Debug) главного меню, соответственно.

Стандартная панель инструментов

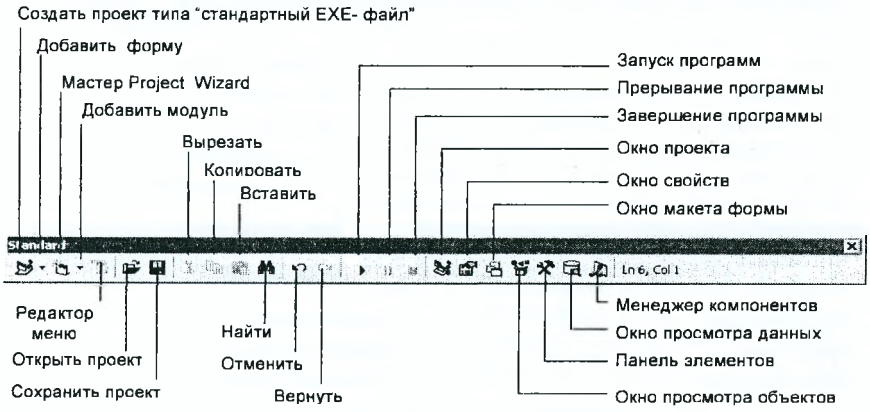


Рис. 4.3. Панель инструментов Стандартная

Стандартная панель инструментов (рис. 4.3) обычно выведена в рабочее окно по умолчанию. Она содержит команды, дублирующие основные команды меню. В этой же панели выводится положение курсора, при работе в окне проекта (Code): указывается номер строки (Ln 6) и номер колонки (Col 1). При работе с формой выводятся размеры формы, например, 4800 x 3600.

Панель редактирования

Панель редактирования (рис.4.4) используется при вводе и редактировании текста программы. Она содержит кнопки для вывода всплывающих списков свойств и методов объекта, констант, синтаксиса для процедур и методов, а также содержит кнопки для управления редактированием текста.

Панель отладки

Панель отладки (рис. 4.5) служит для отладки программ в процессе ее выполнения и обеспечивает запуск программы на выполнение, временную остановку (пауза), выход из программы, установку точек останова, пошаговое выполнение программы, вычисление выделенного выражения.

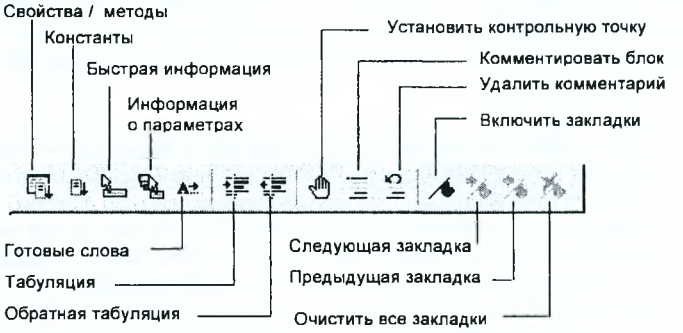


Рис. 4.4. Панель инструментов редактирования

Панель редактора форм

Панель редактора форм (рис. 4.5) применяется при разработке форм. Она позволяет

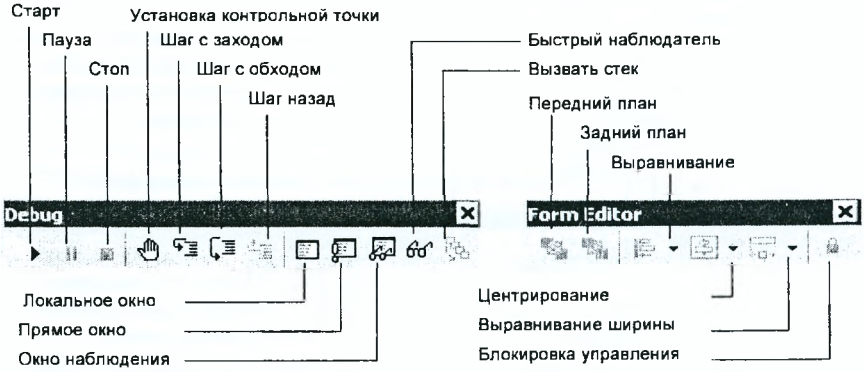


Рис. 4.5. Панели инструментов Отладка и Редактор форм

Панель редактора форм (рис. 4.5) применяется при разработке форм. Она позволяет изменять порядок размещения элементов управления, выравнивать их по горизонтали и вертикали, выравнивать размеры выделенных элементов управления по ширине и высоте, а также блокировать или разблокировать элементы управления в форме.

Панель инструментов элементов управления и компонентов пользователя (Toolbox)

Панель Toolbox содержит элементы управления. Элементы управления – это объекты, которые используются при разработке интерфейса[6] создаваемых приложений. Общее количество доступных к использованию элементов управления зависит от того, какая версия VB используется.

Для добавления новых компонентов к панели инструментов Toolbox из числа зарегистрированных необходимо:

- ввести команду ***Project, Components***, выбрать закладку ***Controls***;
- найти в списке нужный элемент управления и установить напротив него флажок;
- выйти из окна диалога, щелкнув кнопку Ok.

Форма

Создаваемые в Visual Basic окна называются формами. Форма – это основное окно интерфейса разрабатываемой программы, форма - это также основа для создания окон диалога. При загрузке Visual Basic в рабочем окне открыта одна форма, которой по умолчанию присваивается наименование Form1 (рис. 4.2). Форма имеет строку заголовка, кнопку системного меню и кнопки управления развертыванием и свертыванием окна. На ней размещаются все необходимые элементы управления, для удобства размещения которых на форму выводится сетка. В простейшем случае проект может содержать одну форму. Форма настраивается так, как ее будет видеть пользователь при выполнении программы. Перемещают окно формы и изменяют его размеры с помощью мыши во время настройки или программным путем в процессе выполнения программы.

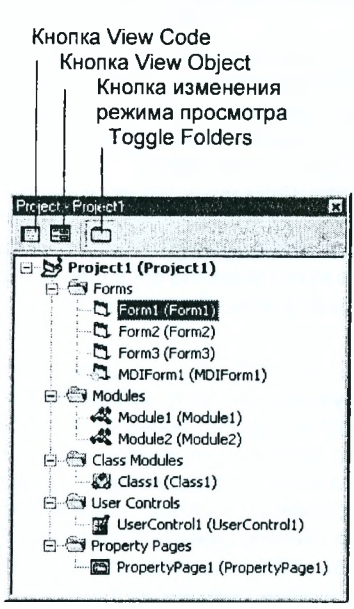


Рис.4.6. Окно Проект

Окно проекта

В окне проекта (броузер проектов) (рис. 4.6.) отображаются все элементы приложения: формы, модули, классы и т.п., сгруппированные по категориям. В VB все разрабатываемые приложения называются проектами. Проект представляет группу связанных файлов и может содержать несколько групп компонентов (формы, модули и т.д.). Все проекты VB строятся по модульному принципу, поэтому и текст программы состоит не из одного большого файла, а из нескольких частей - процедур. Несколько проектов также

[6] Интерфейс – от латинского interface (связь), средство общения пользователя с программой

могут объединяться в группы. Ниже заголовка окна проекта размещены три кнопки: кнопка просмотра текста программы (ViewCode); кнопка просмотра объекта, формы, MDI-формы, управляющего элемента, страницы свойств (View Object) (модули и классы не имеют визуальных компонентов); кнопка изменения режима просмотра (Toggle Folders): в виде списка компонентов или в виде дерева групп компонентов.

Все элементы проекта сохраняются как отдельные и независимые файлы. Поэтому их можно в любое время загружать и сохранять. Это дает возможность использовать в проекте формы и тексты программ, созданные для других проектов, что экономит рабочее время.

Типы файлов проекта Visual Basic приведены в таблице 4.1.

Таблица 4.1 Типы файлов Visual Basic

Тип файла	*Расширение*
Проект Visual Basic (Project)	.VBP
Форма Visual Basic (Form)	.FRM
Программный модуль (Module)	.BAS
Модуль класса (Class Module)	.CLS
Пользовательский элемент управления (User Controls)	.CLT
Файл формы документа ActiveX (Property Pages)	.DOB
Группа проектов	.VBG

Файлы форм сохраняются с расширением .FRM. Содержимое окна проекта сохраняется в специальном файле с расширением .VBP. Файл проекта содержит список элементов, которые надо загрузить в среду разработки. При сохранении проекта сохраняются и все его компоненты. Если несколько файлов проектов объединяются в группу, то их имена сохраняются в файле с расширением .VBG.

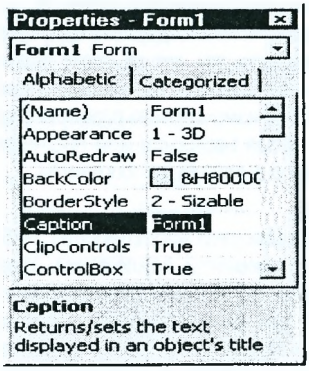


Рис. 4.7. Окно Свойства

Для добавления в проект нового элемента, необходимо вызвать команду ***Project, Add*** ..., а для удаления элемента нужно выделить его в окне проекта и затем выбрать команду меню ***Project, Remove.***

Окно свойств

В окне свойств (рис. 4.7.) задаются свойства выбранного элемента управления. В строке заголовка окна свойств рядом с текстом Properties указывается имя формы, которой принадлежит элемент управления. Поле со списком под строкой заголовка позволяет выбрать требуемый элемент управления. В списке, расположенном ниже, перечислены свойства данного элемента управления. Эти свойства могут быть упорядочены в алфавитном порядке (Alphabetic) или расположены по кате-гориям (Categorized). Набор свойств зависит от класса элемента управления.

Как установить свойства объекта? Имеется несколько способов:

- ввести значение в поле справа от свойства;
- выбрать из списка, который открывается щелчком мыши по полю;

- установить с помощью окна диалога. При щелчке мышью по полю появляется кнопка (...) – троеточие или эллипсис. При щелчке по кнопке троеточие появляется окно диалога для настройки соответствующего свойства;
- загрузить из файла через стандартное окно диалога.

Окно программы

Сразу после запуска окно Программа не отображается. Текст программы в VB разделяется на части – процедуры и записывается в процедуры обработки событий или процедуры пользователя, поэтому он, как правило, связан с определенным элементом управления. Это позволяет открыть окно диалога двойным щелчком по соответствующему элементу формы или по самой форме. Кроме того, окно программы можно открыть щелчком по кнопке *View Code* в окне *Project*.

В верхней строке окна программы (рис.4.8) располагается строка заголовка, в которой указано имя проекта и управляющие кнопки (системного меню, закрытия окна, свертывания и развертывания окна). Ниже строки заголовка расположены два раскрывающихся списка: список объектов (левый) и список процедур (правый).

Список объектов содержит все объекты текущей формы. В их число входит и специальный объект *General*, содержащий текст программы, используемый всеми процедурами формы.

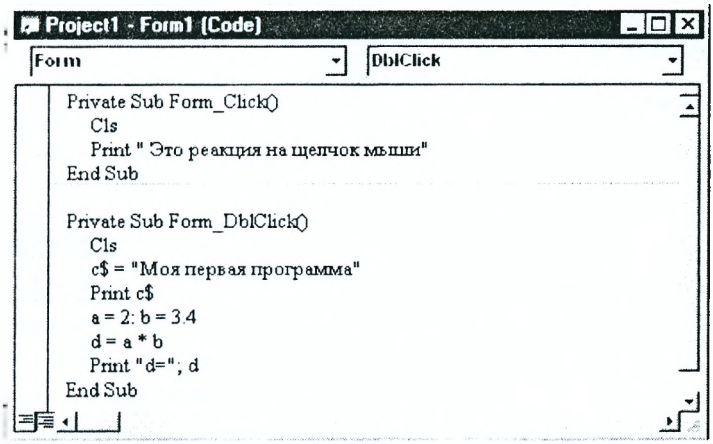


Рис. 4.8. Окно Программы (Code)

Список процедур содержит список всех событий, распознаваемых текущим объектом. Если для объекта уже написаны процедуры обработки событий, то они выделяются здесь жирным шрифтом. Если выбрать любой пункт данного списка, то VB выведет соответствующую процедуру либо ее шаблон в окно программы и поместит в нее курсор.

Окно позиционирования формы

Данное окно позволяет визуально установить место размещения формы при запуске программы и просмотреть, как это будет выглядеть на экране.

4.1.3. РАБОТА С ВНЕШНИМИ УСТРОЙСТВАМИ

Сохранение информации и открытие файлов

По окончании работы проект и форму необходимо сохранить на рабочем диске. Для сохранения информации на диске следует выбрать команду ***File,Save Project*** или ***File,Save Project As.*** Если проект сохраняется первый раз, то сначала сохраняется форма, а затем сам проект. После ввода команды на экране появляется окно диалога (рис. 4.9). Откройте раскрывающийся список "Папка" и выберите в нем необходимый диск, например R:, выберите двойным щелчком мыши нужную папку, введите в строке ввода "Имя файла" имя вашего файла и щелкните по кнопке ***Сохранить***. Имя файла следует выбирать осмысленно, чтобы оно отражало содержание хранимой информации. В этом случае его легче будет отыскать на диске. Расширение имени файла предлагается по умолчанию. Как уже упоминалось, проект приложения сохраняется с расширением VBP, а форма – с расширением FRM.

Учитывая, что проект может содержать большое число различных файлов рекомендуется для каждого проекта создавать отдельную папку.

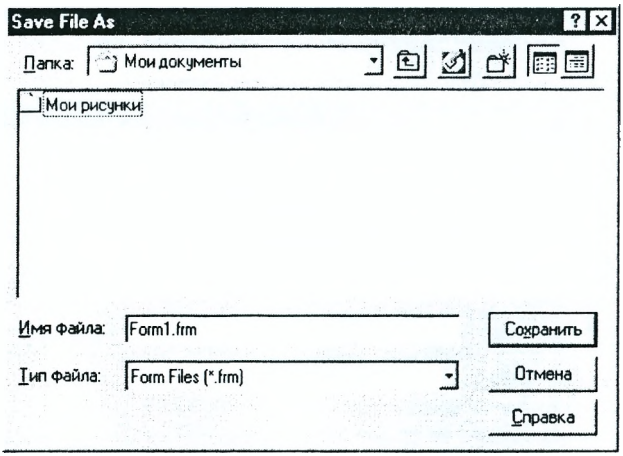


Рис. 4.9. Окно диалога для сохранения файлов

Не рекомендуется *применять другие расширения имен файлов, так как при открытии окна диалога для открытия файлов* ***Open Project*** *они не попадут в список файлов, формируемый по умолчанию.*

Чтобы загрузить в программу уже существующий файл, используется команда ***File,Open Project.*** Окно диалога открытия файлов аналогично окну диалога сохранения файлов, но имеет две закладки: *Existing* (существующий), которая позволяет найти любой файл на компьютере, и *Recent* (недавний), позволяющая загрузить недавно использовавшиеся файлы. Откройте список папок и выберите нужный диск и папку (каталог), выберите двойным щелчком мыши файл для запуска или выделите файл и щелкните по кнопке ***Открыть***.

Команда ***File, Make Project.exe*** позволяет создать исполняемый файл с расширением .EXE.

Вывод информации на печать

Для вывода текста программы на печать служит команда ***File, Print***. После ввода команды появляется окно диалога (рис. 4.10). Это окно позволяет вывести на печать текущую форму, тексты процедур формы, модуль, с которым работает программист или все формы и модули приложения. Переключатели определяют, печатать ли код выделенной части приложения (Selection) , текст программы активной в настоящий момент формы (Current Module) или текст программы всего проекта (Current Project), соответственно. Три флажка (Form Image, Code и Form As Text) определяют, выводить ли на печать изображение формы (то, что пользователь видит на экране), текст программы формы - программных модулей или сведения о свойствах формы и элементов управления, установленных на ней. По умолчанию на печать выводится текст программных модулей (установлен флажок Code).

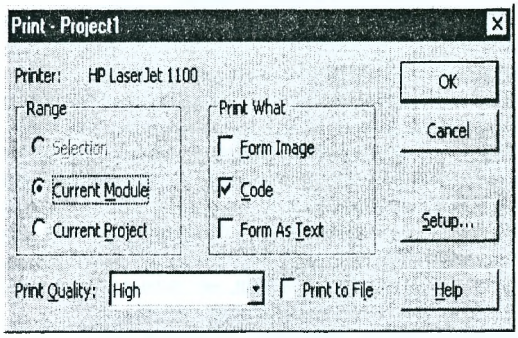


Рис. 4.10. Окно диалога для вывода информации на печать

Команда ***File, Print Setup*** выводит окно с настройками печати. Например, можно выбрать принтер или ориентацию бумаги.

Для организации вывода информации на печать VB взаимодействует с программами Windows более низкого уровня. Для вывода на печать изображения формы со всеми видимыми элементами управления в VB достаточно одной команды – ***PrintForm***.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие версии языка программирования Visual Basic вам известны?
2. Перечислите требования к установке VB.
3. Как запустить программу VB?
4. Назовите основные окна среды разработки программы.
5. Как добавить новые элементы управления на панель инструментов Toolbox?
6. Для чего предназначено окно Проект?
7. Каково назначение окна Свойства?
8. Для чего предназначено окно Программа?
9. Как вызвать окно Программа?
10. Где записывается текст программы объекта?
11. Как сохранить программу на диске?
12. Как загрузить существующую программу?
13. Как вывести на печать текст программы?
14. Как вывести на печать сведения об установленных свойствах формы?

ЗАКЛЮЧЕНИЕ

Что нового узнали мы в этом разделе:

- научились запускать программу Visual Basic 6.0;
- познакомились с составом элементов рабочего окна Visual Basic;
- изучили назначение окон интегрированной среды разработки программы;
- научились сохранять файлы на диске;
- научились выводить текст программы и внешний вид формы на печать.

4.2. ОСНОВНЫЕ ПОНЯТИЯ ОБ ОБЪЕКТНО-ОРИЕНТИРОВАННОМ ПРОГРАММИРОВАНИИ

4.2.1. ОБЩИЕ ПРИНЦИПЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Объектно-ориентированное программирование (ООП) - вид программирования, при котором используется понятие "класс" и связанные с ним объекты.

Техника ООП выгодна тем, что позволяет многократно и эффективно использовать созданные пользователем или другими программистами программные компоненты. Поддержка ООП в Visual Basic базируется на использовании модулей классов.

Основными принципами объектно-ориентированного программирования являются *инкапсуляция, наследование, полиморфизм.*

Инкапсуляция (сокрытие) - сокрытие от пользователя структур данных, процедур и функций для их обработки. Для пользователя объект представляется как "черный ящик". Пользователь знает входные и выходные данные, а что и как происходит внутри объекта ему не известно. Обращение к данным и функциям выполняется через вызов соответствующих методов или свойств. Сокрытие информации позволяет разработчику объекта изменять внутренние принципы его функционирования, не оказывая при этом никакого влияния на пользователя объекта.

Наследование – способность создавать классы, зависящие от других классов. Цель наследования сделать текст программы проще, используя уже готовые объекты и их свойства. Новый объект можно определить на основе уже существующих объектов. При этом новый объект может содержать те же свойства и методы, которые были у исходного объекта (родителя), а также новые свойства и методы. То есть свойства и методы родителя "наследуются" новым объектом. Таким образом образуется иерархия классов. Классы, находящиеся на нижних уровнях могут использовать все свойства и методы, определенные в классах вышележащих уровней. Механизм наследования позволяет переопределять и дополнять данные и методы их обработки. Возможность изменения функциональности метода или свойства обусловлена принципом полиморфизма.

Полиморфизм (многоликий) – способность объекта реагировать на запрос (вызов метода) сообразно своему типу, при этом одно и то же имя свойства или метода может использоваться для различных классов объектов. Суть его заключается в том, что при написании программы, посылающей объекту сообщение, не нужно знать, к какому классу принадлежит этот объект. Все что нужно – это имя сообщения и его параметры. Например, оператор + может использоваться как для строковых так и для числовых переменных. Однако это будут совершенно разные операции. В первом случае будут объединяться строки символов, во втором – вычисляться сумма числовых переменных. Или другой пример: метод Print формы и метод Print объекта Debug. Первый выводит информацию на форму, а второй в окно непосредственного наблюдения, хотя синтаксис у них одинаковый: <имя_объекта>.Print <список выражений>. Программисту не надо будет задумываться, как будет работать программа при выводе информации на форму или в окно непосредственного наблюдения. Программа сама определит, что надо делать в соответствии с классом объекта.

Ключевыми понятиями ООП являются *класс, объект, свойство, метод, событие.*

Класс - это абстрактный тип объекта, шаблон или план, по которому создаются объекты определенного класса. Класс содержит внутренние переменные, недоступные пользователю, внешние переменные, доступные пользователю через свойства класса, внутренние (закрытые) и внешние (открытые) процедуры и функции. Внутренние процедуры и функции недоступны пользователю, а внешние процедуры и функции доступны пользователю в виде методов и обработчиков событий.

На основе классов создаются объекты. Каждый *объект* – это экземпляр определенного класса. Например, есть *класс форма*, при загрузке на основе этого класса VB автоматически создает и помещает в конструктор форм экземпляр этого класса - объект Form1, для которого пользователь может установить требуемые свойства из набора уже известных свойств объекта класса форма. Или другой пример, есть *класс текстовое поле*, помещая на форму текстовое поле, мы создаем объект типа текстовое поле. Поместив на форму пять текстовых полей, мы создадим тем самым пять экземпляров объектов класса текстовое поле. На основе данного класса могут создаваться подклассы, используя механизм наследования. VB 6.0 *не поддерживает* в полной мере механизм наследования. Однако он позволяет создавать модули классов. Модули классов предоставляют свою функциональность в виде объектов. Модули классов могут использоваться как внутри данного проекта, так и могут оформляться в виде ActiveX-элементов, доступных для использования в других проектах. Например, можно создать модуль класса "транспортные средства", на основе которого можно создавать подклассы "легковые автомобили", "грузовые автомобили" и т. д., а на основе них создавать объекты: грузовые автомобили, легковые автомобили и т. п. (рис. 4.11).

Класс - это абстрактный тип объекта, шаблон или план, по которому создаются объекты определенного класса. Класс содержит свойства, методы и события, общие для всех объектов данного класса

Объект – совокупность свойств (параметров) определенных сущностей и методов их обработки (программных средств). Объект содержит инструкции (программный код), определяющие действия, которые может выполнять объект, и обрабатываемые данные.

Объект – это экземпляр класса, характеризующийся определенным набором свойств, методов, событий.

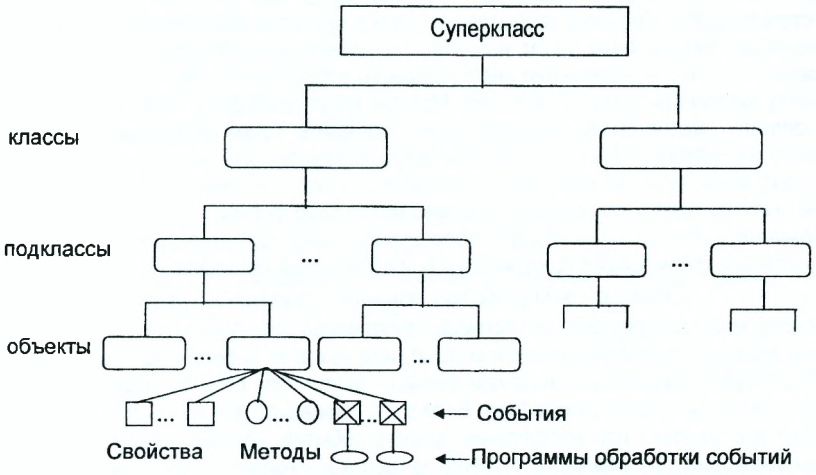


Рис. 4.11. К понятию класса объектов

Метод – программа действий над объектом или его свойствами. Метод рассматривается как программный код, связанный с определенным объектом, и который осуществляет преобразование свойств, изменяет поведение объекта.

Методы – это функции или процедуры, определенные в классе.

Методы могут быть объявлены как глобальные (внешние, открытые) или как локальные (внутреннние, закрытые). Глобальные процедуры, определенные в классе, работают так же, как и обычные подпрограммы и функции, но они доступны только пользователям объекта. Методы, объявленные как глобальные, образуют программный интерфейс класса, и программы работают с ними без всяких ограничений. Методы объявленные как локальные используются классами для выполнения внутренних операций, недоступных за пределами класса. Локальные методы объясняют суть принципа инкапсуляции. Программист может изменить локальную подпрограмму или функцию, определенную в классе, и это никак не отразится на его программном интерфейсе, доступном пользователю.

Методы классов могут пересекаться. Например, метод Print формы выполняет те же действия, что и метод Print объекта Picture (рисунок). В то же время метод Print объекта Printer выполняет другие функции – выводит данные на принтер, а не на форму. Здесь проявляется принцип полиморфизма – реагировать сообразно типу объекта.

Свойство – характеристика объекта, его параметр. Все объекты наделены определенными свойствами, которые в совокупности выделяют объект из множества других объектов, т.е. придают ему качественную определенность. Свойства объектов разных классов, так же как и методы, могут пересекаться.

Свойство – характеристика объекта, его параметр.

Свойства могут быть открытыми (внешними) и закрытыми (внутренними). Открытые свойства могут использоваться процедурами не входящими в данный класс (например, свойство Text объекта Text). Они образуют часть интерфейса класса. Закрытые свойства используются для хранения информации, не входящей в интерфейс. Они предотвращают намеренное или случайное изменение данных, обеспечивающих нормальную работу класса.

Чем отличается метод от свойства? Свойство определяет внешний вид объекта и его местоположение. Свойства задаются на этапе разработки приложения, а также могут изменяться программным путем, например, положение объекта определяется значением свойств Top и Left (положение левого верхнего угла объекта). Методы – это команда объекту выполнить какие-то действия. Методы могут приводить к тем же результатам, например, метод Move при выполнении программы также перемещает объект в требуемое положение – Move X,Y. Методы могут использоваться не только при выполнении программы, но и на этапе проектирования, а некоторые методы можно вызвать даже не выделив память под объект, например метод Load формы.

Событие – это свойство объекта процедурного типа. События позволяют классу обмениваться информацией с приложением при соблюдении определенных условий.

Событие – это свойство объекта процедурного типа

События могут генерироваться системой – внутренние события или пользователем – внешние события. Например, щелчок мышью, или нажатие клавиши на клавиатуре – внешние события; загрузка или выгрузка формы – внутренние события. Каждому событию объекта соответствует ***процедура***. Если в эту процедуру поместить программу, то она будет выполняться при наступлении данного события. Использование механизма события избавляет программиста от необходимости многократной проверки значения некоторой величины до тех пор, пока не будет выполнено некоторое условие. Цикл активного опроса поглощает ресурсы процессора и замедляет выполнение программы.

Например, изменение значения текстового поля – есть событие Change. Для обработки этого события предусмотрена процедура

```
Private Sub Text1_Change ()    - ' заголовок процедуры
. . .
End Sub                        - ' конец процедуры
```

Шаблон этой процедуры входит в состав программного кода объекта TextBox. Предположим, что на форму установлен объект TextBox, имя объекта – Text1. Если в окне программы в списке объектов (рис. 4.8) выделить объект Text1, а в списке процедур выбрать событие Change то в окно программы автоматически вставляется шаблон этого события.

Чтобы эта процедура работала, в нее надо поместить текст программы, например:

```
Private Sub Text1_Change ()
    Длина=Val(Text1.Text)
End Sub.
```

Тогда при каждом изменении значения свойства Text объекта Text1 переменной Длина будет присваиваться новое значение.

Взаимосвязь указанных выше понятий представлена на рис.4.11.

Программа, созданная с помощью инструментальных средств ООП, содержит объекты с их характерными свойствами, для которых разработан графический интерфейс пользователя. Как правило, работа с приложением осуществляется с помощью *экранных форм* с объектами управления - *окон диалога*. Экранные формы[7] используются также для выполнения заданий и перехода от одной части программы к другой. При разработке приложений, для объектов управления уточняется перечень событий и создается пользовательский метод обработки – текст программы на языке программирования в виде событийных процедур.

В объектно-ориентированном программировании используется следующий синтаксис операторов для обращения к методам и свойствам объектов:

обращение к методу:

ИмяОбъекта.Метод

ИмяФормы.ИмяОбъекта.Метод

изменение свойства объекта в процессе выполнения программы:

ИмяОбъекта.Свойство ="значение"

использование свойства объекта для присвоения значения переменной:

Переменная = ИмяОбъекта.Свойство

Например:

Form1.Show - показать форму. Form1 - имя объекта, Show – метод.

Text1.Text ="Учебное пособие" - свойству Text объекта Text1 присваивается значение "Учебное пособие". Text1 – объект, Text – свойство объекта, "Учебное пособие" – значение свойства Text.

Художник = Text2.Text - переменной "Художник" присваивается значение свойства Text объекта Text2.

---

[7] Формы, представленные на экране монитора, не имеют материального аналога, они существуют только в виде программ на диске или в оперативной памяти компьютера, а также в виде визуального изображения на экране монитора. Поэтому их можно называть "экранные формы". В дальнейшем по тексту будем писать просто "формы".

Свойства объектов

Каждый объект характеризуется определенным набором свойств. Свойства придают объекту качественную определенность, выделяют его из совокупности других объектов. Некоторые из свойств объекта характерны для всех объектов, другие присущи только данному объекту. Общими свойствами являются такие свойства, как имя, надпись, положение объекта и его размеры, цвет фона, цвет текста, индекс, индекс обхода.

При выделении объекта его свойства отображаются в окне свойств (Properties). Однако в окне свойств отображаются только те свойства, которые могут быть установлены пользователем на этапе разработки проекта. При выделении группы объектов в окне свойств отображаются только общие свойства, которые присущи всем выделенным объектам.

Имя элемента (Name). Каждому элементу управления имя присваивается по умолчанию при установке его на форму, например, форме – *Form*, метке – *Label*, текстовому полю – *Text* и т.д. Так как однотипных объектов может быть много, то программа нумерует их: *Label1*, *Label2* и т.д. Однако такие имена мало информативны, поэтому каждому объекту необходимо присвоить оригинальное, достаточно информативное имя. Полям ввода можно присваивать имена в соответствии с элементом данных: длина, ширина, успеваемость. Кнопкам - в соответствии с их функциональным назначением: стоп, пуск, пауза. Имена объектов используются в программе для обращения к ним, поэтому имя объекту необходимо присваивать до написания программы. Имя объекта необходимо писать латинскими символами. В имени объекта не допускается использование пробелов.

Примите за правило: создал объект – присвой ему имя.

Рекомендуется имя объекта начинать с префикса. Например, для формы – frm, для текстового поля – txt, для метки – lbl, для кнопки – cmd и т.д.: frmForm1, txtDlina, lblDlina, cmdStart. При этом префикс начинается со строчной буквы, а базовое имя (Dlina) - с прописной буквы. При таком подходе можно использовать одинаковые имена, например, для метки и окна ввода. Программа различит их, так как префиксы у них разные. Использование префиксов в имени объекта облегчает анализ программы при поиске ошибок и поиск в списке объектов окна программы.

Надпись (Caption) – это текстовое сообщение, которое выводится на соответствующий объект в форме. Каждому объекту программа автоматически присваивает имя и аналогичную надпись. Надпись на элементе управления может совпадать с именем объекта. В отличие от имени, надпись можно писать на любом языке, т.к. она не используется в программе. Надпись на объекте можно создавать в процессе разработки программы, а также при ее выполнении. Текст программы в этом случае представляет собой строку следующего вида:

```
ИмяФормы.ИмяОбъекта.Свойство ="<текст>"
frmForm1.cmdExit1.Caption = "Резервная кнопка"
```

свойству Caption кнопки cmdExit1 формы frmForm1 присваивается значение "Резервная кнопка".

Положение объекта. Положение объекта в форме определяется координатами верхнего левого угла (рис. 4.12). Для этого используется два свойства:

Top – расстояние от верхнего края формы и **Left** – расстояние от левого края формы. Все расстояния задаются в "твипах ". *Твип* – экранно-независимая единица измерения. Один твип равен 1/20 точки принтера. Это гарантирует независимость отображения элементов приложения от разрешения дисплея.

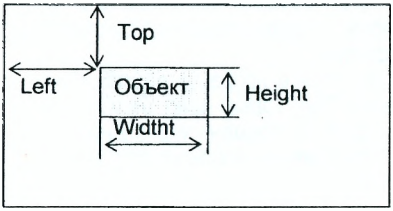


Рис.4.12. Положение объекта в форме

Размеры объекта определяют также два свойства: **Height** и **Width** (высота и ширина).

Цвет. Управление цветовым оформлением элементов осуществляется с помощью свойства **BackColor**, **FillColor** и **ForeColor**. Этим свойствам по умолчанию назначаются стандартные цвета Windows.

Свойство **BackColor** определяет цвет фона. При проектировании цвет выбирают из палитры в диалоговом окне настройки, а в программе цвета задаются либо с использованием цветовой схемы RGB, либо константами библиотеки VBRUN (см. раздел Графические средства Visual Basic).

Свойство **ForeColor** определяет цвет текста. Для формы оно определяет также цвет сетки.

Свойство **FillColor** определяет цвет заполнения рисованных объектов.

Appearance – позволяет изменять внешний вид объекта, имеет два значения: Flat – плоский; 3D – объемный.

Enabled – доступность элемента управления. Может принимать два значения: *True* – истина и *False* - ложь. Если значение свойства установлено в *False*, то элемент управления будет недоступен пользователю. Такой элемент управления при выполнении программы подсвечивается блеклым серым цветом. Данное свойство используется для управления объектами. Например, кнопка "Расчет", может быть отключена до тех пор, пока не будут введены все исходные данные.

Font - это свойство позволяет оформлять шрифты, используя все возможности Windows.

Index – индекс. Определяет номер элемента в массиве элементов управления.

TabIndex – индекс обхода. Присваивается программой автоматически при помещении элементов управления на форму в порядке их создания. Этот индекс определяет порядок перемещения *фокуса* по элементам управления при нажатии клавиши *Enter*. Порядок обхода может быть изменен пользователем на этапе разработки программы. Если объект выделен, то говорят, что на него установлен фокус. Фокус может устанавливаться автоматически в порядке обхода элементов управления, в соответствии с возрастанием значения свойства TabIndex, или устанавливаться программным путем.

ToolTipText – позволяет ввести текст, который отображается в подсказке, появляющейся при зависании указателя мыши на элементе управления.

Visible – видимость элемента. Это свойство также имеет два значения: *True* и *False*. Если установлено значение False, то элемент управления будет невидим.

Методы объектов

Объекты имеют различное число методов, например, объект типа надпись имеет 10 методов, объект типа форма – 22 метода. Общими для этих объектов являются методы **Move** – перемещать, **OLE Drag** - перемещать с использованием механизма OLE (вставка и внедрение), **Refresh** – перерисовать объект, и **SetFocus** - установка фокуса.

Одним из важных понятий при обращении к элементам управления в Windows является понятие ***фокус.*** При нажатии клавиш на клавиатуре управление получает активный элемент, то есть элемент, имеющий фокус. Например: при нажатии клавиши Enter выполняются те команды, которые связаны с командной кнопкой, имеющей фокус; текст вводится в поле ввода, на которое установлен фокус, и т.д. Не все объекты могут получать фокус, например Надпись. Если элемент получает фокус, то он выделяется особым образом, например, текстовое поле отмечается мигающим курсором, командная кнопка - пунктирной рамкой по пери-

метру и т. д. С установкой и потерей фокуса связаны два события и один метод: **GotFocus** – событие возникающее при установке фокуса; **LostFocus** – событие, связанное с потерей фокуса, **SetFocus** – метод, обеспечивающий установку фокуса.

Фокус на элемент управления может быть установлен в процессе разработки программы путем присвоения свойству **Default** значения True. Только один элемент управления на форме может иметь значение Default. Фокус на элемент управления может быть установлен также и в процессе выполнения программы с помощью метода **SetFocus** командой следующего вида:

<имя объекта>.SetFocus

например:

txtText1.SetFocus

События объектов

Внешние события объектов связаны с мышью и клавиатурой.

Click. Событие Click вызывается, как только пользователь выполнит щелчок мышью на элементе управления.

DblClick. Событие DblClick вызывается двойным щелчком мыши на элементе управления. Временной интервал между двумя щелчками устанавливается в панели управления Windows;

KeyDown – нажатие клавиши[8];

KeyUp - отпускание клавиши;

KeyPress – возвращает код ASCII нажатой клавиши.

При нажатии и отпускании клавиши возвращаются два параметра: *KeyCode* и *Shift*. Системная переменная KeyCode содержит клавиатурный код нажатой клавиши, а Shift – информацию о нажатии клавиш Shift, Ctrl или Alt. Например:

Control_KeyUp(KeyCode As Integer, Shift As Integer)

После нажатия клавиши события наступают в такой последовательности: KeyDown, Key Press, KeyUp.

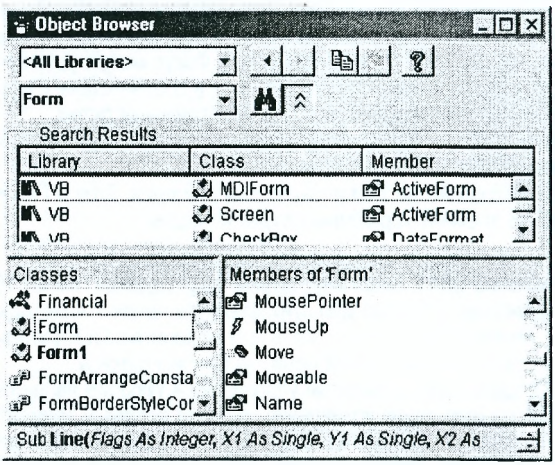


Рис.4.13. Окно диалога для просмотра объектов

KeyPreview. Это свойство присуще только форме. Оно определяет порядок обработки событий, связанных с нажатием клавиш. Если свойству KeyPreview формы присвоить значение True, то событие, связанное с клавиатурой, передается сначала форме, а затем текущему элементу управления.

GotFocus – событие, возникающее при установке фокуса.

LostFocus – событие, возникающее при потере фокуса.

[8] Подробнее о событиях KeyDown, KeyUp,KeyPress см. раздел 4.11.2

Окно просмотра объектов

Все свойства, события и методы объекта можно просмотреть в окне просмотра объектов *Object Browser* (рис.4.13.), который содержит каталог объектов. Для вывода окна диалога необходимо ввести команду ***View, Object Browser***. Выбрать в этом меню в верхнем поле списка необходимую библиотеку либо объекты из всех библиотек. Стандартные элементы управления содержит библиотека VB (Visual Basic objects and procedures). В списке *Classes* перечислены все классы объектов VB. Здесь *Form* - класс объекта типа Форма, *Form1* – объект, экземпляр данного класса. *Financial* – класс типа Модуль, остальные элементы – константы объекта Form1. ***Константы*** - это данные, значения которых не меняются при выполнении программы. Для каждого объекта имеется свой набор констант.

После выбора объекта в списке *Members of* выводятся все свойства и методы, относящиеся к выбранному объекту, в данном случае к форме. В этом окне отображаются свойства (MousePointer, Moveable, Name), события (MouseUp) и методы (Move). Свойства обозначаются характерной пиктограммой серого цвета - карточка с рукой. События выделяются желтым цветом, методы - зеленым цветом. Нажав клавишу F1 или кнопку "? ", можно вызвать справочную информацию. Чтобы найти конкретное свойство или метод, можно воспользоваться кнопкой поиска (бинокль). Введите во второй строке ввода имя объекта, например Form, и щелкните по кнопке поиска. После окончания поиска в окне результаты поиска (Search Results) появится имя библиотеки и искомый объект, а в окне Members – свойства и методы этого объекта.

В нижней части окна дается краткое описание выделенного объекта. В примере на рис. 4.13 выведено сведение о методе Line.

4.2.2. БАЗОВЫЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

Создание Windows-приложений практически невозможно без использования элементов управления, так как они позволяют пользователю взаимодействовать с этими приложениями. Набор таких элементов управления не ограничен и может расширяться за счет так называемых пользовательских элементов управления (custom controls).

В данном разделе рассматриваются три элемента управления, которые позволяют создавать простейшие приложения: Командная кнопка (CommandButton), Надпись (Label) и Текстовое поле или Поле ввода (TextBox).

Командная кнопка

Кнопка используется для управления процессом: начало, окончание, прерывание и т. д.

Основными свойствами являются имя, название, положение, размеры, цвет, доступность, видимость.

Дополнительно можно указать следующие свойства:

Cancel - обеспечивает перехват нажатия клавиши Esc и вызов события Click для соответствующей кнопки. Например, если свойству Cancel кнопки cmdEnd присвоить значение True, то нажатие клавиши Esc вызовет закрытие формы. Только у одной кнопки значение свойства Cancel может быть равно True.

Default – определяет, является данная кнопка активной по умолчанию или нет. Свойство имеет два значения: *True* и *False*. По умолчанию – False. Если установлено значение True, то фокус установлен на данной кнопке, то есть кнопка активна. Нажатие клавиши Enter перехватывается и передается этой кнопке. Так же как и у свойства Cancel, только у одной кнопки значение свойства Default может быть равно True.

Style – определяет стиль оформления, также имеет два значения: Standard – стандартный и Graphical – графический. В графическом режиме можно изменять цвет кнопки и помещать на кнопку рисунки с помощью свойства Picture.

Надпись

Надпись предназначена для отображения текста, который пользователь не может изменить с клавиатуры. Но текст можно изменять программным путем. Из особенных свойств можно выделить следующие:

Alignment – выравнивание текста. Имеет три значения: 0- выравнивание по левому краю, 1 - по правому краю, 2 - по центру;

AutoSize - автоматическое приведение ширины объекта в соответствие с длиной текста. Если свойство AutoSize равно False и длина вводимого текста больше ширины надписи, то текст усекается. Если свойство AutoSize равно True, то размер объекта приводится в соответствие с длиной текста. При этом значение свойства WordWrap должно быть равно False.

WordWrap - автоматический перенос длинного текста на другую строку. При установке значения этого свойства в True длинный текст будет автоматически переноситься на новую строку независимо от значения свойства AutoSize. Обычно свойства WordWrap и AutoSize используются совместно. Пример взаимодействия этих свойств приведен на рис. 4.14.

Номер варианта	Результат	Значение свойства	
		WordWrap	AutoSize
1	Это пример совместного	False	False
2	Это пример совместного использования свойств WordWrap и AutoSize	False	True
3	Это пример совместного	True	False
4	Это пример совместного использования свойств WordWrap и AutoSize	True	True

Рис.4.14. Влияние свойств WordWrap и AutoSize на изменение размеров надписи

В первом варианте длинный текст усекается в соответствии с размерами элемента управления. Во втором варианте размер надписи приводится в соответствие с длиной текста. В третьем варианте текст автоматически переносится на следующую строку, но этого не видно, так как высота надписи автоматически не изменяется. В четвертом варианте текст автоматически переносится на следующую строку и увеличивается высота надписи в соответствии с размером текста.

Для получения требуемого результата в четвертом варианте установите вначале свойство WordWrap в True, а затем значение свойства AutoSize также установите в True.

Текстовое поле

Текстовое поле является основным элементом управления, предназначенным для ввода и вывода данных.

Основные свойства текстового поля совпадают с перечисленными выше, но есть и особенные свойства:

Text – аналог свойства Caption. Через это свойство осуществляется ввод данных в программу и вывод данных на экран;

MultiLine – определяет, может ли поле содержать более одной строки текста. Обычно совмещается с установкой свойства ScrollBars.

ScrollBars – вывод линеек прокрутки. Свойство Scrollbars позволяет устанавливать горизонтальную (1), вертикальную (2) или обе (3) линейки прокрутки. Если свойству ScrollBars присвоено значение 0 (None), то линейки прокрутки в окно не выводятся.

В текстовом поле можно выделять и заменять текст. Это осуществляется программным путем с помощью свойств *SelStart*, *SelLength*, *SelText*.

SelStart – начало выделения.

SelLength – длина выделяемого текста.

SelText – замена текста.

Например:

```
Text1.SelStart=2          'начать выделение со второго символа
Text1.SelLength=6         'выделить шесть символов
Text1.SelText = "Привет"  'заменить выделение на слово "Привет"
```

MaxLength – определяет максимальное число символов – По умолчанию – 0, что означает максимальное значение – 32 тысячи символов.

PassWordChar – позволяет заменять вводимые символы звездочками. Это свойство используется для ввода пароля.

Locked – запрещает пользователям изменять содержимое поля. Поле можно просматривать, но нельзя редактировать или удалять. Однако значение поля можно изменить программным путем.

Объект TextBox может обрабатывать 23 события. Основные события текстового поля связаны с вводом данных: Click, DblClick, KeyDown, KeyUp, GotFocus, LostFocus, которые рассмотрены были ранее;

Дополнительно можно отметить свойство **Change** - изменение значения текстового поля.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимается под объектно-ориентированном проектированием, в чем его преимущество перед обычным программированием на других языках высокого уровня, например, языке Qbasic, Pascal?
2. Поясните основные принципы объектно-ориентированного программирования.
3. Дайте определение понятиям класс, объект, свойство, событие, метод.
4. Что такое объект? Какова его структура?
5. В чем отличие свойств от методов?
6. Перечислите основные события объектов.
7. Перечислите основные методы объектов.
8. Поясните назначение командной кнопки CommandButon?
9. Перечислите свойства, события и методы командной кнопки.
10. Для чего предназначена Надпись (Label)?

11. Перечислите свойства, события и методы Надписи.
12. Поясните назначение окна ввода TextBox.
13. Перечислите свойства, события и методы окна ввода.
14. Как просмотреть свойства объекта?
15. Как просмотреть все элементы объекта (свойства, методы, события, константы)?
16. Приведите синтаксис присвоения значения объекта переменной.
17. Приведите синтаксис присвоения объекту значения переменной.

ЗАКЛЮЧЕНИЕ

В данном разделе мы познакомились:

- с основными принципами объектно-ориентированного проектирования. К ним относятся инкапсуляция – сокрытие внутренних парамеров и методов их преобразования от пользователя; наследование – способность создавать классы, зависящие от других классов. Свойства и методоы родительских объектов досупны всем объктам, созданным на их основе; полиморфизм – способность объекта реагировать на вызов метода сообразно своему классу;
- с понятиями класс, объект, свойство, метод, событие;
- с основными свойствами объектов Надпись, Окно ввода, Командная кнопка.

4.3. НАЧАЛО РАБОТЫ В VISUAL BASIC

В предыдущих разделах мы ознакомились с основными элементами управления, их событиями, свойствами и методами, необходимыми для разработки приложений. Но для того, чтобы разработать приложение, которое выполняло бы какую-нибудь полезную функцию, необходимо знать как ввести данные в программу и как вывести полученные результаты на экран или печать.

4.3.1. ВВОД ДАННЫХ

Данные в программу могут быть введены несколькими способами:

- присвоением начальных значений переменным непосредственно в программе с помощью оператора ***Let***;
- вводом данных с клавиатуры в режиме диалога с помощью элемента управления ***TextBox***;
- с помощью диалогового окна ***InputBox***.

Присвоение начальных значений переменным

Присвоение начальных значений переменным осуществляется с помощью оператора ***Let***. Этот оператор не обязателен и может быть опущен.

Синтаксис оператора:

[Let] <имя_переменной> = <выражение>

Например:

```
Let a=3.754:  b=Sin(x)+exp(x): Let c$= "Зимние каникулы": c1$="Новый год"
```

Ввод данных с помощью элемента управления TextBox

В режиме диалога пользователя с программой данные можно ввести с клавиатуры непосредственно в строку ввода элемента управления TextBox. Текстовое поле используется для ввода самой разнообразной информации. При вводе данных с клавиатуры в активное текстовое поле программа не делает различий между буквами и цифрами: все данные вводятся как текст. Поэтому для перевода текста в числа и обратного перевода чисел в текст, используются функции преобразования строковых переменных:

Val(C) – преобразование текста в число ;

Str(N) – преобразование числа в строковую пременную или **Str$(N)** - для преобразования в символьную переменную переменных типа Variant.

```
<переменная_символьного_типа>= txtText1.Text
<переменная_числового_типа>= Val(txtText1.Text)
```

Например:

```
Длина = Val(Text1.Text)   'присвоение числовой переменной А значения
                          ' свойства Text объекта Text1
Text2.Text = Str$(Длина)  'присвоение свойству Text объекта Text2 значения
                          ' числовой переменной
```

Ввод данных с помощью окна диалога InputBox

В режиме диалога данные можно ввести в программу с помощью функции ***InputBox*** (рис.4.15.). Функция InputBox используется тогда, когда необходимо ввести только краткую информацию. Она имеет следующий синтаксис:

<Переменная> = InputBox (prompt [,title] [,default] [,xpos] [,ypos] [,helpfile], context])

Параметр: *Prompt* определяет текст, отображаемый в диалоговом окне как приглашение ("Введите значение переменной X").

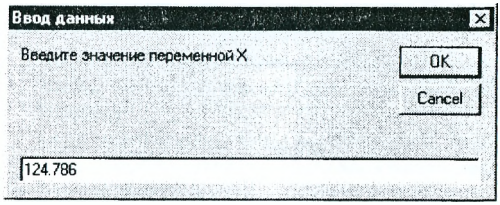


Рис. 4.15. Окно диалога функции InputBox

Title – отвечает за надпись заголовка ("Ввод данных"). Если этот параметр не указан, то отображается название приложения.

Default – определяет значение по умолчанию, отображаемое в строке ввода (124.786).

Параметры *XPos* и *YPos* используются совместно и указывают координаты верхнего левого угла окна. По умолчанию окно отображается посредине экрана.

Функция InputBox имеет еще два необязательных параметра *HelpFile* и *Context*, которые позволяют открывать определенные файлы справочной системы. При нажатии кнопки OK, функция InputBox_возвращает строку, введенную пользователем. При нажатии кнопки Cancel возвращается пустая строка.

Например:

```
x=val(InputBox("Введите значение переменной X",124.786, "Ввод данных")
```

В приведенном примере:

- "Введите значение переменной X" – текстовое сообщение, выводимое в окно ввода;
- 124.786 – значение переменной, принимаемое по умолчанию;
- "Ввод данных" - текст, выводимый в строку заголовка;
- значение координат XPos и YPos принимается по умолчанию, то есть окно ввода будет выведено по середине экрана.

Из-за громоздкости окна ввода его рекомендуется использовать только при отладке программы. В готовой программе рекомендуется использовать текстовое поле TextBox.

4.3.2. ВЫВОД ДАННЫХ

Ввод данных осуществляется несколькими способами:

- с помощью оператора *Print*;
- с помощью *текстового поля*;
- с помощью окна сообщений *MessageBox*;
- с помощью *сетки MSFlexGrid*.

Оператор Print

Оператор Print может выводить информацию непосредственно в форму или графический объект PictureBox:

Print ["текстовое сообщение"] [;/_/,] <список выражений> [;/,]

Синтаксис оператора Print рассмотрен в разделе 3.2.2:

Для позиционирования точки вывода используются свойства ***CurrentX*** и ***CurrentY*** объекта, а также функции ***Tab(N) и Spc(N)***. Например:

```
CurrentX=100: CurrentY=50
Print x,y
```

или

```
Print Tab(100); x; Spc(10); y
```

Свойства CurrentX и CurrentY перемещают точку вывода в указанные координаты X и Y. Расстояния задаются в твипах.

Функции Tab и Spc используются в операторе Print. Они 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 формы, а функция Spc – относительно текущей позиции (рис. 4.16).

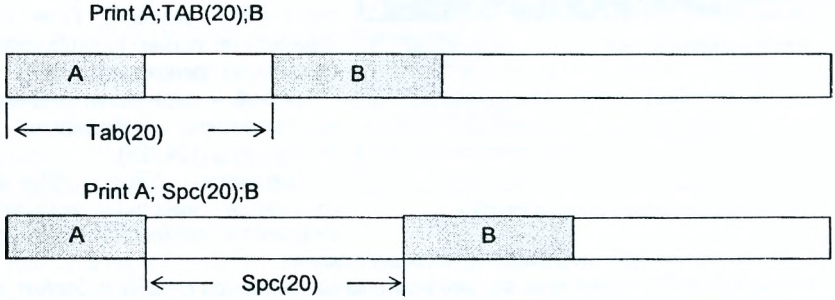


Рис. 4.16. Использование функций Tab и Spc

Текстовое поле TextBox

Текстовое поле TextBox может использоваться для вывода информации в любом месте формы. Позиция элемента управления может устанавливаться как при разработке формы, так и в процессе выполнения программы, например:

```
Text1.Top=Y:  Text1.Left=X
Text1.Text = "<выводимый текст>"
```

Окно диалога MessageBox

Окно MessageBox подобно окну InputBox и служит для вывода информации. Этот компонент широко используется для вывода различных сообщений в приложениях Windows.

MessageBox можно вызвать как команду и как функцию.

Синтаксис команды:

```
MsgBox Prompt [, Buttons] [,Title] [, Helpfile, Context]
```

Синтаксис функции:

```
<Переменная >= MsgBox (Prompt [, Buttons] [,Title] [, Helpfile, Context])
```

Параметры *Prompt, Title, Helpfile, Context* имеют то же значение, что и в функции InputBox. В отличие от синтаксиса команды в синтаксисе функции аргументы заключаются в скобки.

Параметр *Buttons* определяет состав кнопок в окне. Он формируется из нескольких частей:

Buttons = Button + Icon + Default + Modal +Extras

где *Button* – количество кнопок и их состав;

Icon – вид пиктограммы;

Default – определяет, какая кнопка активна по умолчанию;

Modal – вид диалогового окна (окно приложения или системы). Модальность означает, что выполнение приложения возможно только после закрытия окна;

Extras – дополнительные свойства.

По умолчанию выводятся кнопки ОК и Отмена.

Значения категорий параметра Buttons приведены в табл. 4.2

Код, возвращаемый функцией, зависит от того, какая кнопка нажата пользователем:

1 – OK, 2 – Cancel, 3 – Abort, 4 – Retry, 5 – Ignore, 6 – Yes, 7 – No

Это позволяет анализировать код нажатой клавиши и управлять программой в зависимости от реакции пользователя.

Пример 4.1. Приветствие (рис.4.17):

```
MsgBox "Здравствуй, товарищ!", vbExclamation, "Приветствие".
```

Функция:

```
КодВозврата=MsgBox ("Конец игры?", vbCritical, "Игра")
If КодВозврата=2 Then End ' Если КодВозврата равен 2 То-
гда Выход из программы
```

Пример 4.2. Формирование параметра Buttons.

```
Type=4+32+256+1
MsgBox "Сообщение", Type, "Заголовок".
```

При данном коде в окне отображаются кн. Yes и No (код 4), пиктограмма Warning Query (код 32), активна кнопка No (код 256), окно модальное (код 1).

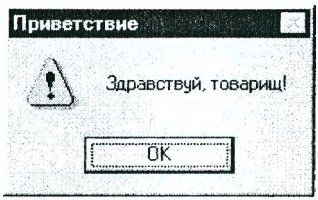


Рис.4.17. Окно MsgBox

Пример 4.3. Выведите на экран результаты вычисления X и Y.

Если текст длинный или желательно разместить результаты вычислений в несколько строк, используется константа Visual Basic ***VbCr*** – возврат каретки. Для сложения символьных переменных используется символ & или +.

```
MsgBox "X=" & Str(x) & VbCr & "Y=" & Str(y)
```

Таблица 4.2 Константы параметра Buttons

Код	*Константа*	*Значение*
Константы категории Button		
0	(vbOkOnly)	Выводится только кнопка ОК
1	(vbOkCancel)	Выводится кнопка ОК и Cancel
2	(vbAbortRetryIgnore)	Выводится кнопка Abort, Retry, Ignore
3	(vbYesNoCancel)	Выводится кнопка Yes, No, Cancel
4	(vbYesNo)	Выводится кнопка Yes, No
5	(vbRetryCancel)	Выводится кнопка Retry, Cancel

Таблица 4.2 Константы параметра Buttons

Код	*Константа*	*Значение*
Константы категории Icon		
16	VbCritical	Отображает пиктограмму критического сообщения (Critical Message) 
32	VbQuestion	Отображает пиктограмму Warning Query (запрос) 
48	VbExclamation	Отображает пиктограмму Warning Message (предупреждение) 
64	vbInformation	Отображает пиктограмму Information Message (информация) 
Константы категории Default		
0	vbDefaultButton1	по умолчанию активна первая кнопка
256	vbDefaultButton2	по умолчанию активна вторая кнопка
512	vbDefaultButton3	по умолчанию активна третья кнопка
768	vbDefaultButton4	по умолчанию активна четвертая кнопка
Константы категории Modal		
1	vbModal	модальное диалоговое окно приложения;
4096	vbSystemModal	модальное диалоговое окно системы
Константы категория Extras		
16384	vbMsgBoxHelpButton	дополнительная кнопка для справки
65536	vbMsgBoxSetForeground	отображение диалогового окна в фоновом режиме
524288	vbMsgBoxRight	текст выровнен по правому краю
1048576	vbMsgBoxRtlReading	текст отображается справа налево

Пример 4.4. Решение квадратного уравнения

Найти корни квадратного уравнения $ax^2 + bx + c = 0$. Для управления проектом поместите на форму кнопку Command1. Ввод данных в программу выполнить с помощью функции InputBox. Результат вывести с использованием функции MsgBox.

Порядок работы:

- Поместите на форму командную кнопку:

- щелкните мышью по значку кнопки на панели ToolBox (кнопка выделяется светлым тоном);

- переместите указатель мыши на форму, нажмите левую клавишу мыши и, протягивая мышь, установите требуемые размеры кнопки;

- Откройте окно программы двойным щелчком мыши по форме и выберите объект Command1;

- Откройте список объектов и выберите свойство Click этого объекта. Если щелкнуть дважды мышью по кнопке Command1, то сразу попадаете в обработчик события Click кнопки;

- Подготовьте контрольные данные для тестирования программы.

- Запишите в обработчик события Click текст программы:

```
Private Sub Command1_Click( )
   a = Val(InputBox ("Введите значение коэффициента a"))
   b = Val(InputBox ("Введите значение коэффициента b"))
   c = Val(InputBox ("Введите значение коэффициента c"))
   D = Sqr(b^2-4*a*c)
   If  D<0 then MsgBox  "Нет Действительных корней" ,vbCritical
   x1 = (- b + Sgr(D))/(2*a) :    x2 = (- b - Sqr(D))/(2*a)
   MsgBox "x1 = " & Str(x1) & vbCr & "x2 = " & Str(x2),,"Корни уравнения:"
End Sub
```

- Запустите программу;
- Щелкните мышью по кнопке Command1 для выполнения расчетов.
- Проверьте правильность работы программы.
- Сохраните программу на диске.

Внимание: Для получения подсказки по встроенным функциям VB6 откройте окно Object Browser, введите в строке поиска имя функции, например, Sin и щелкните по кнопке Поиск или откройте библиотеку VBA, класс Math и найдите в списке Members of Math требуемую функцию.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие функции используются для перевода строковых переменных в числовые?
2. Какие функции используются для перевода числовых переменных в строковые?
3. Какими способами осуществляется ввод данных?
4. Как присвоить значение полю TextBox? Приведите формат команды присвоения значения свойству Text объекта Text1 программным путем.
5. Приведите синтаксис функции InputBox?
6. Приведите синтаксис оператора Print и поясните назначение разделителей.
7. Приведите синтаксис команды MsgBox.
8. Чем отличается синтаксис функции MsgBox от синтаксиса команды MsgBox?
9. Как разместить данные в окне MsgBox в несколько строк?
10. Какие значения возвращает функция MsgBox и как они могут использоваться?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы изучили:

- способы ввода данных. Данные вводятся с помощью оператора Let, текстового ввода TtxtBox, функции InputBox;
- способы вывода результатов вычисления. Данные могут выводиться непосредственно в форму или объект PictureBox с помощью оператора Print, в текстовое окно TextBox, с помощью команды или функции MsgBox;
- ознакомились с назначением функций Val(C), Str$(N), Str(N) и константой vbCr.

4.4. ПРОГРАММИРОВАНИЕ В VISUAL BASIC

4.4.1. ОСНОВНЫЕ ПОНЯТИЯ О ПРОГРАММИРОВАНИИ В СРЕДЕ VB

Среда программирования

Структура проекта

Все программы, работающие в среде Windows, принято называть ***приложениями***. Каждое приложение, разрабатываемое в среде VB, называется ***проектом***. Проект

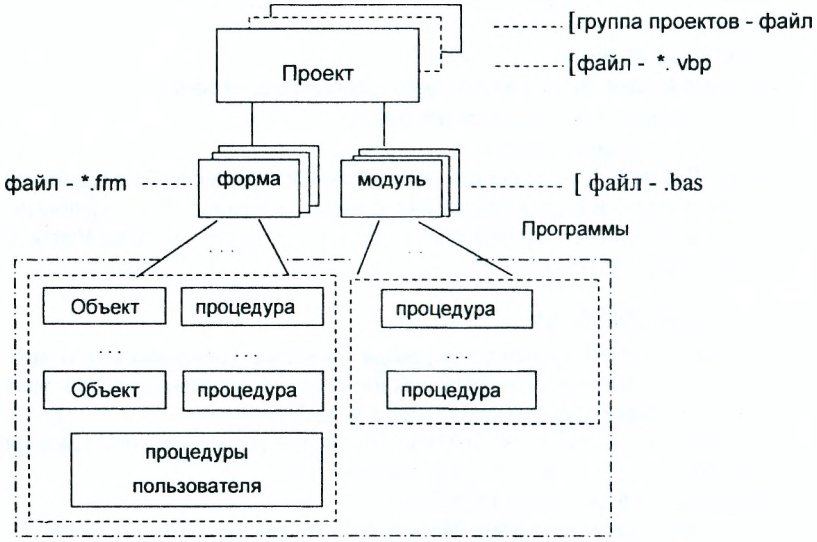


Рис. 4.18.Структура проекта

состоит обычно из форм и модулей, с которыми связаны тексты программ. Текст программы проекта состоит из множества подпрограмм, оформленных в виде процедур обработчиков событий и пользовательских процедур (рис. 4.18.).

Проекты, формы и модули сохраняются в отдельных файлах и могут самостоятельно загружаться и добавляться к другим проектам, с помощью команд ***Project, Add Project; Project, Add Form*** и ***Project, Add File***.

Текст программы, применяемый во всех формах, можно разделить на несколько модулей, хранимых раздельно.

Структура проекта отображается в окне *Project* в виде древовидной структуры (рис. 4.6, 4.18).

Когда в VB начинается подготовка больших проектов, очень остро встает вопрос о повторном использовании текста программы. По этой причине целесообразнее хранить процедуры и функции в отдельных модулях, чем подключать их к форме.

Различают стандартные модули, модули классов и модули пользовательских элементов управления.

В ***стандартных модулях*** размещается текст программ, доступ к которым должен иметь весь проект. Эти программы применяются для повторного использования. *Стан-*

дартные модули не имеют визуальных компонентов. Новый модуль создается при помощи команды ***Project, Add Module***. Для загрузки существующего модуля необходимо ввести команду ***Project, Add Module*** и далее открыть закладку ***Existing*** или использовать команду ***Project, Add File***. По умолчанию файлы стандартных модулей имеют расширение ***.bas.***

Окно Программы

Окно Программы (рис. 4.19) – часть среды разработки VB. Здесь вводится и редактируется текст программы, который позволяет программе реально выполнять какую-либо работу. Эту часть среды разработки проектаVB с полным основанием можно назвать ***средой программирования***.

Каждая форма и модуль имеет собственное, связанное с ней окно программы, которое открывается двойным щелчком мыши по форме или по имени файла в окне проекта. В окне программы расположены два раскрывающихся списка: список объектов и список свойств и процедур. ***Список объектов*** содержит список всех элементов, расположенных на форме. ***Список свойств*** содержит список всех процедур обработчиков событий выделенного объекта и процедур данной формы.

Текст программы формы или модуля состоит из программы раздела Главная (General) и программ обработчиков событий, пользовательских процедур и функций, разделенных пунктирной линией. Эта линия формируется автоматически при создании новой процедуры. Раздел Главная служит для объявления типов переменных и написания пользовательских процедур и функций, доступных всем процедурам формы.

На рис. 4.19 в разделе Главная записан один оператор – Option Explicit, а ниже записаны коды обработчиков событий Click и Load формы. Разделительная полоса ***Split Bar*** (ее маркер находится над вертикальной полосой прокрутки) служит для деления ок-

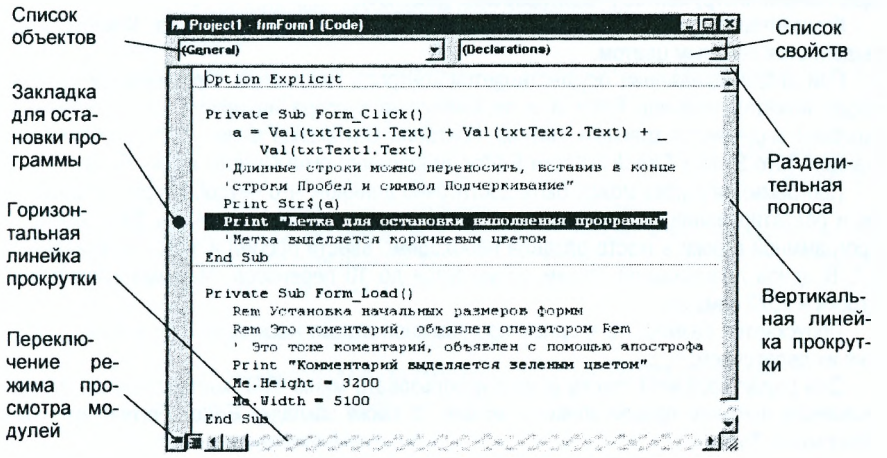


Рис. 4.19. Окно кода

на редактирования на две части по горизонтали. В этом случае в каждом окне можно просматривать разные участки программы, повышается удобство отладки и написания программы. Установка и удаление разделительной полосы осуществляется с помощью

мыши. Для установки разделительной полосы зацепите ее мышью и перетащите в нужное место экрана. Чтобы удалить разделительную полосу передвиньте ее мышью к верхнему краю окна.

Две кнопки слева от горизонтальной полосы прокрутки служат для переключения режима просмотра модулей. Первая кнопка (Procedure View) устанавливает режим просмотра одного модуля, вторая (Full Module View) – просмотр нескольких модулей. Последний режим установлен по умолчанию.

Для редактирования текста можно использовать все стандартные функции редактирования Windows (выделение текста, копирование, удаление, вставка). Кроме того, редактор текста программы VB обеспечивает:

- контроль вводимого кода. Размер кода не должен превышать установленные размеры (64 Кбайт). Выводится всплывающее окно с информацией о текущем объекте;
- выдачу информации о синтаксисе текущего оператора VB (Быстрая информация - Quick Info). При вводе ключевого слова, за которым следует пробел или открывающаяся скобка, на экране появляется подсказка, где приведен синтаксис данного оператора;
- вывод на экран списка всех свойств и методов текущего объекта, после того как введена точка в конце названия объекта (List Properties, Methods). Методы выделяются при этом зеленым цветом;
- вывод на экран списка типов переменных после ввода служебного слова As при объявлении типов переменных;
- получение списка возможных констант (Available Constants) после ввода знака равно после имени объекта.

Указанные выше функции можно включить или отключить с помощью команды ***Tools, Options***. После ввода команды выберите закладку ***Editor*** и установите или снимите флажки ***Auto Quick Info*** и ***Auto List Members*** в группе ***Code Setting*** или с помощью панели инструментов Редактирование (рис. 4.4).

Комментарии вводятся с помощью оператора ***Rem*** или апострофа. Комментарий выделяется зеленым цветом.

При вводе выражений осуществляется контроль синтаксиса. При наличии ошибок после нажатия клавиши Enter или перемещения курсора на другую строку строка с ошибкой выделяется красным цветом. Контроль синтаксиса может быть отключен командой ***Auto Syntax Check*** вкладки ***Editor*** диалогового окна ***Options*** меню ***Tools***.

Программная строка может быть достаточно длинной, что неудобно при ее просмотре и редактировании, а также при выводе текста программы на печать. Для переноса программной строки в месте раздела необходимо ввести пробел и знак подчеркивания "_". В одной программной строке допускается до 10 переносов. Максимальная длина строки – 1023 символа.

Допускается размещать в одной программной строке несколько операторов, разделяя их двоеточием.

Для редактирования текста можно использовать меню ***Edit***, контекстное меню, вызываемое щелчком правой клавиши мыши, а также закладку ***Editor Format*** окна ***Options*** меню ***Tools***.

VB имеет мощные *средства отладки*. Отметим некоторые из них:

- установка контрольных точек. Щелкните мышью по рамке окна напротив оператора, где вы хотите остановить программу, - на рамке появляется метка и вся строка выделяется коричневым цветом. Для отмены метки щелкните по ней мышью. Метка не устанавливается на пустую строку и на строку комментария;

• после остановки программы можно просмотреть значения всех интересующих вас переменных и тем самым установить причину прерывания программы или неправильной ее работы. Значения переменных высвечиваются при зависании на них мыши.

В синтаксисе языка Visual Basic используются операторы, функции, переменные и константы.

Операторы – синтаксические конструкции, которые управляют вычислительным процессом. Операторы образуют текст программы.

Функции – вычислительные процедуры, предназначенные для выполнения наиболее часто используемых вычислительных или логических операций.

Переменные – именованные области памяти, предназначенные для хранения данных. Значения переменных могут изменяться в процессе выполнения программы.

Константы – постоянные величины, или поименованные области памяти, предназначенные для хранения данных. Значения констант не изменяются в процессе выполнения программы.

Переменные

Имя переменной

По определению переменная – это именованная область памяти, предназначенная для хранения данных. Поэтому каждой переменной присваивается идентификатор – имя. Имя переменной начинается с буквы и может содержать до 255 символов. В имени не допускаются пробелы, нельзя использовать также символы "." и "&". Имя переменной рекомендуется начинать с односимвольного ***префикса*** (табл. 4.2), характеризующего тип хранимых данных. Тип переменной устанавливается при объявлении переменной или определяется программой автоматически по содержанию присвоенной ей информации. Тип переменной задает определенный формат и размер содержимого переменной.

Чтобы не было путаницы с типами данных и проблем при поиске ошибок, рекомендуется обязательно объявлять тип переменной.

Типы переменных

VB имеет большое число типов переменных – 14. Основные типы приведены в табл. 4.2., все типы переменных приведены в приложении 2. Не допускается использование в программе двух одинаковых имен, отличающихся только типом.

Особо необходимо отметить переменные типа Variant. Если тип переменной не указан, она будет объявлена программой как Variant. В переменной типа Variant можно хранить переменные любого типа. Преобразование типов переменных в этом случае осуществляется автоматически. Но при этом в некоторых случаях могут возникать ошибки. Кроме того, использование переменной типа Variant приводит к перерасходу оперативной памяти, а также этот тип данных неприемлем для использования в качестве аргументов для тех функций, параметры которых описаны явным образом. Поэтому рекомендуется с самого начала объявлять типы переменных явно.

Объявление типов переменных явно считается хорошим стилем программирования.

Способы объявления переменных

Тип переменной можно объявить несколькими способами:

1. Неявно с помощью специального символа – ***суффикса***, записанного после имени переменной (табл. 4.6). Спецсимвол следует указывать только при первом использовании переменной:

```
C$ = "текст": a! = 1.769: B% = 12674
```

Таблица 4.6. Типы переменных

Суффикс	*Префикс*	*Тип переменной*	*Объявление по умолчанию*	*Занимаемая память в байтах*	*Описание*
$	s	String	DefStr	1 байт на символ	Строковая переменная переменной длины, 2 миллиарда символов для динамически изменяемых срок
%	n	Integer	DefInt	2	Целое, одинарной длины (-32768,+ 32768)
&	l	Long	DefLng	4	Целое, двойной длины (от - 2 147 483 648 до + 2 147 483 647)
!	f	Single	DefSng	4	Вещественное, одинарной точности, +-(10^{45} ... $3*10^{38}$)
#	d	Double	DefDbl	8	Вещественное, двойной точности, 1E127, +- ($5*10^{324}$... $1,8*10^{308}$)
-	b	Boo-lean	DefBool	2	Логическое, используется для представления логических переменных, имеет два значения : True и False
-	v	Date	DefDate	8	Дата и время, от 1/1/100 до 12/31/ 9999; от 0:0:0 до23:59:59
-	-	Object	-	4	Экземпляр класса; объект типа OLE
-	-	Variant	DefVar	16 + 1 байт/сим-вол	Вариантный

2. По умолчанию с помощью операторов вида ***DefType***. Этот оператор используется только в разделе Главная. Синтаксис оператора:

DefType <список символов >

Например:

```
DefStr C
DefInt I-L
```

В данном примере первый оператор объявляет, что все переменные, имя которых начинается с символа "С", являются переменными строкового типа, второй оператор объявляет, что все переменные, имена которых начинаются с символов I, J, K, L, являются переменными целого типа одинарной точности.

3. Явное объявление с помощью операторов ***Dim, Private, Static, Public, Global***. При объявлении переменной указывается ее имя и тип. Синтаксис операторов объявления переменных:

Dim <имя_переменной> As <тип_переменной> [,<имя_переменной> As <тип_переменной >]

> ***Внимание***: *Для каждой переменной ее тип объявляется отдельно. В одном операторе Dim можно объявлять несколько переменных, разделяя их запятыми и указав для каждой переменной ключевое слово As и тип переменной.*

Например:

```
Dim a As Integer, b As Single
Dim nCount As Integer, I As Integer, J As Integer
Publuc sText As String, NameFile As String
Global dFiz As Double
```

> *Соглашение о типах переменных, принятых по умолчанию, можно изменить, используя суффикс или оператор Dim..*

Контроль типов переменных

С целью исключения ошибок, связанных с неправильным объявлением имен переменных, рекомендуется объявлять все переменные явно. Контроль типов переменных можно установить с помощью оператора ***Options Explicit***. Этот оператор необходимо

поместить в разделе «Главная» (General) данной формы. Если переменная не была объявлена, то при запуске программы выдается сообщение об ошибке.

Другой способ обеспечить контроль типов переменных – установить флажок в опции ***Require Variable Declaration*** на вкладке ***Editor*** в меню ***Tools , Options***. В этом случае оператор ***Options Explicit*** будет автоматически вставляться в раздел «Главная» формы при ее создании.

Область определения (видимости) переменных

В VB есть три вида областей определения, характеризующих доступность или видимость переменной: локальная; контейнера (формы или модуля); глобальная.

Локальная переменная доступна только в текущей процедуре или функции. Локальные переменные объявляются оператором ***Dim***.

Переменная контейнера доступна только в текущей форме, модуле или классе. Объявляются оператором ***Dim*** в разделе Главная (General) формы или модуля. Переменная модуля объявляется оператором ***Private***.

Глобальные переменные доступны во всем проекте. Они объявляются в разделе Главная главного модуля программы оператором ***Global*** или ***Public*** (файл с расширением .bas).

Время жизни переменных

Переменные, объявленные как локальные, при выходе из процедуры удаляются из памяти, а при новом вызове инициализируются заново.

Переменные уровня формы инициализируются при первом открытии формы и сохраняют свои значения до окончания работы проекта. С целью экономии оперативной памяти переменные уровня формы могут быть уничтожены при закрытии формы командой ***Set <имя формы>=Noting***.

Глобальные переменные при выходе из программы сохраняют свои значения до окончания работы проекта.

Статические переменные

В некоторых случаях необходимо, чтобы при выходе из процедуры переменные сохраняли свое значение. Например, при использовании процедуры для печати страниц номер страницы должен увеличиваться при каждом вызове процедуры. Поэтому при выходе из процедуры переменная, хранящая значение текущего номера страницы, должна сохранить свое значение, чтобы при последующем вызове процедуры можно было увеличить номер страницы на единицу.

С целью обеспечения сохранности значений переменных при выходе из процедуры они могут быть объявлены в данной процедуре как статические оператором ***Static***:

```
Static nPageNumber As Integer
```

> *Статические переменные являются локальными для процедуры, в которой они используются, но их значения сохраняются до повторного вызова процедуры.*

Чтобы объявить статическими все переменные процедуры ключевое слово Static следует записать в заголовке процедуры:

```
Static Sub <имя_процедуры>(аргументы)
Static Sub ПечатьФормы (x )
```

Константы

Основное отличие констант от переменных состоит в том, что их значения нельзя изменить в процессе выполнения программы. Они всегда сохраняют значение, присвоенное при разработке. Области видимости для констант определяются так же как и для

переменных. Константы бывают локальные, уровня контейнера и глобальные. При объявлении констант используется ключевое слово ***Const***. Глобальная константа объявляется как ***Public Const***.

Глобальные константы объявляются только в модуле.

Одновременно с объявлением константе можно присвоить и значение:

```
Public Const <Имя константы>=<Значение>
Private Const <Имя константы>=<Значение>
```

Например:

```
Public Const  Pi = 3.1415926538897932
Private Const  G = 9.81
Public Const nName = «Лев Толстой»
Const ПлотностьМатериала = 2.25
Const Масса = ПлотностьМатериала * Высота* Ширина* Длина
```

Информацию о константах, их значениях и применении можно получить, обратившись к соответствующим разделам справки или воспользовавшись каталогом объектов: войти в меню ***View, Object Browser***, щелкнуть кнопку ***Object Browser*** на панели инструментов или нажать клавишу ***F2.***

Константы можно объявлять и с указанием типа данных:

```
[ Public/Private] Const <Имя константы> As <Тип_данных>=<Значение>
    Public Const  Pi As Double = 3.1415926538897932
```

Для указания типа данных используются те же ключевые слова, что и при объявлении переменных.

Visual Basic имеет большое число встроенных констант, буквально "на все случаи жизни". В виде констант в VB определены коды цветов, коды клавиш, флажки, константы, задающие тип пиктограмм и набор кнопок в окне сообщения Message Box и др..

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните структуру проекта в Visual Basic.
2. В чем состоит отличие формы от модуля?
3. Расскажите назначение окна кода и его элементов.
4. Какие средства редактирования и ускорения ввода программы в окне программы имеет Visual Basic?
5. Приведите синтаксис присвоения значений переменным и свойствам объектов.
6. Назовите основные типы переменных и констант в VB.
7. Какие способы объявления типов переменных применяются в VB?
8. Как обеспечить контроль типов переменных?
9. Какие операторы используются для объявления типов переменных по умолчанию?
10. Поясните, что такое область видимости и время жизни переменных?
11. Как обеспечить сохранность данных при выходе из процедуры?
12. Как можно получить сведения обо всех константах, используемых в проекте, и константах VB?

ЗАКЛЮЧЕНИЕ

В данном разделе мы познакомились:

- со структурой проекта;
- с основными элементами окна программы;
- с понятиями имя, время жизни и область видимости переменных;
- с типами и способами объявления типов переменных;
- с константами и способами их объявления.

4.4.2. ПРОЦЕДУРЫ

Процедура – это часть программы, выполняющая определенную функцию. Процедура имеет один вход и один выход. Войти в процедуру можно только через ее заголовок. Выход из процедуры осуществляется автоматически по ее окончании. Имеется возможность досрочного выхода из процедуры командой ***Exit Sub***.

Различают внутренние процедуры и внешние процедуры. Внутренние процедуры связаны с определенными событиями, и потому их называют чаще всего *обработчиками событий*. Внешние процедуры создаются пользователем.

Внутренние процедуры (обработчики событий)

Синтаксис внутренней процедуры для обработки события имеет следующий вид:

```
[ОбластьОпределения] Sub [ИмяФормы.]Объект_Событие(аргументы)
  <операторы>
End Sub
```

Опции, указанные в квадратных скобках, могут быть опущены. Если аргументы отсутствуют, ставятся пустые скобки. Например:

```
Private Sub frmForm1.cmdVvod_KeyUp(KeyCode As Integer, Shift As Integer)
<операторы>
End Sub
```

В данном примере *ОбластьОпределения* – область доступности или видимости процедуры: *Private* – локальная, доступна только в данной форме или модуле; *Public* – глобальная, доступна во всем проекте. *frmForm1*– имя формы, *cmdVvod* – имя кнопки, *KeyUp* – событие, связанное с нажатием клавиши, *KeyCode* – возвращаемый код нажатой клавиши , *Shift* – возвращаемый код нажатой дополнительной клавиши;

```
Private Sub cmdCommand1_Click ( )
   <Тело процедуры>
End Sub
```

В данном примере имя формы опущено, по умолчанию принимается имя текущей формы, передаваемые параметры – аргументы отсутствуют.

Процедуры обработки событий для каждого объекта выбираются из правого списка окна программы (Code).

Внешние процедуры (процедуры пользователя)

Visual Basic позволяет создавать и пользовательские процедуры. Основной отличительной чертой пользовательских процедур является то, что они не связаны ни с каким событием и вызов их пользователь осуществляет по своему усмотрению. В виде пользовательской процедуры можно оформить любую подпрограмму и использовать ее в текущем проекте или сохранить на диске и использовать в других программах. Для создания пользовательской процедуры необходимо перейти к секции Главная, в окне программы ввести служебное слово ***Sub*** и ***имя процедуры*** и нажать клавишу Enter. После этого появится новая процедура, например:

```
Sub Privet()
    Print "Здравствуй читатель"
End Sub
```

Процедура, описанная в разделе «Главная», доступна всем другим процедурам формы.

Синтаксис процедуры пользователя совпадает, в основном, с синтаксисом процедуры обработки событий:

```
[ОбластьОпределения] Sub ИмяПроцедуры(аргументы)
<Тело Процедуры>
End Sub
```

Вызов процедуры

Для вызова процедуры используются два способа: по имени процедуры и с помощью оператора Call.

При вызове процедуры ***по имени*** после имени указываются без скобок передаваемые параметры:

ИмяПроцедуры аргумент1, аргумент2, ...

Передаваемые параметры по типам должны соответствовать типам переменных, указанных в описании процедуры. В качестве передаваемых параметров могут использоваться строки символов, имена переменных или функций, возвращающих значения заданного типа.

При вызове процедуры с помощью оператор ***Call*** передаваемые параметры заключаются в скобки:

Call ИмяПроцедуры (аргумент1, аргумент2,...)

Оператор Call используется, как правило, для вызова внешних процедур. Чтобы отличить вызов функции от вызова процедуры, рекомендуют не использовать оператор Call для вызова процедур (при вызове функций передаваемые параметры также заключаются в скобки).

Пример 4.5 Процедура вычисления площади поверхности параллелепипеда.

```
' Процедура, объявленная в секции главная формы
Sub SParallel(a As Single, b As Single, h As Single, s As Single)
    s = 2 * (a * b + (a + b) * h)
End Sub
Sub VParallel(a As Single, b As Single, h As Single, v As Single)
    v = a * b* h
End Sub
-----------------------------------------------------------
Private Sub frmForm_Click()
   ' Вызывающая процедура
          Dim x As Single, y As Single, z As Single, s As Single, v As Single
   x = Val(InputBox("Длина параллелепипеда"))
   y = Val(InputBox("Ширина параллелепипеда "))
   z = Val(InputBox("высота параллелепипеда "))
   ' вызов процедуры
   SPoverch x, y, z, s
   Call   Vparallel(x,y,z,v)
   Print "S=";Str(s)
   Print "V=";Str(v)
End Sub
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое процедура? Какие виды процедур используются в Visual Basic?
2. Приведите синтаксис обработчика событий.
3. Приведите синтаксис прльзовательской процедуры.
4. Чем отличается синтаксис процедуры пользователя от синтаксиса обработчика событий?
5. Как вызывается внутренняя процедура?
6. Как вызывается пользовательская процедура?

ЗАКЛЮЧЕНИЕ

В данном разделе мы познакомились с понятием процедура.

Процедуры бывают внутренние и внешние.

Внутрение процедуры – обработчики событий. Они вызываются автоматически при наступлении определенного события.

Внешние процедуры создаются пользователем. Они вызываются оператором Call или просто по имени процедуры. При вызове процедуры оператором Call аргументы заключаются в скобки. При вызове процедуры по имени аргументы указываются без скобок.

4.4.3. ФУНКЦИИ

Встроенные функции

Visual Basic имеет большое число встроенных функций. Можно выделить следующие категории функций:

- математические;
- работы с массивами;
- преобразования типов данных;
- работы с символьными переменными;
- работы с датами и временем;
- финансово-математические;
- работы с файлами;
- управления компьютером;
- обработки ошибок

Многие из этих функций совпадают с функциями языка Basic, но появилось много и новых, специфических функций, расширяющих возможности языка программирования по работе с файлами, обработке данных, работы с датами и временем, финансовые.

Математические функции приведены в разделе 2 табл. 2.1.

Функции форматирования чисел, дат и времени

Для управления представлением чисел, дат и времени используются функции ***Format*** и ***Round.***

Синтаксис функции Round:

Round(<имя_переменной>, n)

где n – число десятичных знаков после запятой. Например:

b=Round(x,2)

переменной b присваивается значение переменной *x* с двумя знаками после десятичной точки.

Debug.Print Round(x, 2)

Синтаксис функции Format:

Print Format (<переменная, значение>, "шаблон")

Например: Print Format (1124.75; "currency")

Для управления выводом информации в функции Format используются *стандартные и пользовательские* шаблоны, приведенные в табл. 4.7 и 4.8. Для управления вы-

Таблица 4.7 Стандартные шаблоны функции Format

Имя шаблона	*Описание*	*Пример*
General Number	Выводит числа без специального форматирования.	1124.75
Currency	Выводят знак денежной единицы, число с разделением тысяч и двумя знаками после запятой (000,000,000.00р.).	1 124.75р.
Fixed	Выводит минимум одну цифру перед запятой и две – после запятой.	1124.75
Standard	То же, что и Fixed, кроме того добавляется разделитель тысяч.	1 1221.75
Percent	Исходные числа умножаются на 100 и добавляется знак процента.	112475.00%
Scientific	Экспоненциальная форма	1,12E+03
Yes/No	Для ненулевых значений возвращается Yes, а для нулевых – No.	
True/False	Для ненулевых значений возвращается True, а для нулевых – False.	
On/Off	Для ненулевых значений возвращается On, а для нулевых – Off.	

Продолжение таблицы 4.7

Примеры использования стандартных шаблонов	
Шаблоны	*Результат*
x = 1124564.9875645	
Debug.Print Format(x, "General Number")	1124564,9875645
Debug.Print Format(x, "Currency")	1 124 564,99р.
Debug.Print Format(x, "Fixed")	1124564,99
Debug.Print Format(x, "Standard")	1 124 564,99
Debug.Print Format(x, "Percent")	112456498,76%
Debug.Print Format(x, "Scientific")	1,12E+06

водом данных могут использоваться также и *дополнительные функции форматирования*, приведенные в табл. 4.9.

Наряду с пользовательскими процедурами имеются и пользовательские функции. Синтаксис пользовательских функции похож на синтаксис пользовательских процедур. В качестве аргументов могут быть константы, переменные или выражения. В конце процедуры записывается оператор *ИмяФункции = результат* , здесь ИмяФункции – простая переменная, имя которой совпадает с именем функции. Этим оператором имени функции передается вычисленное значение.

Использование пользовательских функций

Функции пользователя используются так же, как и встроенные функции Visual Basic. То есть, они используются в выражениях справа от знака равно, могут включаться также в оператор Print:

< имя переменной > = ИмяФункции (аргумент 1 [, аргумент 2, ...])

Print ИмяФункции(аргумент1 [, аргумент2, ...])

Допускается также использование функции в следующем формате:

ИмяФункции (аргумент 1 [, аргумент 2, ...])

Внутри функции можно объявлять локальные переменные, указывая их тип: статические или динамические. Переменные уровня формы доступны всем функциям, подключенным к данному модулю или форме. Преждевременный выход из функции осуществляется с помощью оператора ***Exit Function***.

Подключение пользовательских функций и процедур к текущей форме осуществляется следующим образом:

Таблица 4.8 Пользовательские шаблоны функции Format

Символ	*Назначение*	*Число, шаблон*	*Результат*
0	цифровое знакоместо, отображается цифра или 0	12450023 "000000000"	012450023
#	цифровое знакоместо, отображается цифра или пробел перед значащими цифрами;	12450 "########"	12450
.	место десятичной точки	124500.2375 "#######.##"	124500.24
,	разделитель тысяч	124,500.2345 "#,###,###.##"	124 500.23
%	знак процентов, число будет умножено на 100	0.25 "####.##%"	25%
E-	показательная степень числа, при отрицательном показателе степени перед ним будет отображаться знак «-», знак «+» отображаться не будет	0,0005834 "#.##E-"	5,83E-4
E+	то же, что и предыдущий, только при положительном показателе будет отображаться знак «+».	124500.2375 "#.##E+"	1.25E+5

Функции пользователя

Таблица 4.9 Дополнительные функции форматирования

Функция	*Тип параметра на входе*	*На входе*	*На выходе*
Format Currency	Денежный	8675.309	$8,675.31
Format Numeric	Числовой	-5000	(5,000.00)
Format Percent	Процентный	0,1234	12.34%
Format DataTime	Дата/Время	"12-31 13:34"	12/31/98 1:34:00 PM

Примеры использования дополнительных функций форматирования

```
x = 3456221.345
Debug.Print FormatCurrency(x)                    3 456 221,35р.
Debug.Print FormatNumber(x)                      3 456 221,35
Debug.Print FormatPercent(x)                     345 622 134,50%
Debug.Print FormatDateTime("07-11 10:30")        07.11.04 10:30:00
```

- открыть окно Программы (Code);
- выбрать пункт меню ***Tools, Add Procedure***;
- указать в диалоговом окне ***Имя процедуры***;
- установить переключатель ***Function***;
- щелкнуть по кн. ***OK***.

Пример 4.6. Вычислить значение Sin(x) с точностью 0.001. Математическая модель решения данной задачи представляется следующей формулой:

$$\mathrm{Sin}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^{i+1}\frac{x^{2i-1}}{(2i-1)!} + \ldots = \sum_{i=1}^{\infty}(-1)^{i+1}\frac{x^{2i-1}}{(2i-1)!}$$

Вычисление факториала оформим в виде процедуры. Входным параметром функции будет целое число одинарной длины n – факториал. Функция возвращает целое число двойной длины.

```
Public Function Factorial(n As Integer) As Long
    Dim i As Integer, F As Long
    F = 1
    For i = 1 To n
        F = F * i
    Next i
    Factorial = F
End Function
-------------------------------------------------------
Private Sub Form_Click()
    Dim n As Integer, F As Long, k As Integer
    Dim x As Single, y As Single, s As Single, e As Single
    x = Val(InputBox("Аргумент X"))
    e = Val(InputBox("Точность вычисления"))
    y = 1 + e: s = 0: k = 1
    While Abs(y) >= e
        ' при вычислении значения Y вызывается функция Factorial
        y = (-1) ^ (k - 1) * x ^ (2 * k - 1) / Factorial(2 * k - 1)
        s = s + y
        k = k + 1
    Wend
    Print Str$(s)
End Sub
```

В Visual Basic 6.0 функции имеют новую возможность: они могут возвращать масси-

вы. Можно, например, возвратить массив целых чисел без предварительного преобразования его в строку и обратно:

Пример 4.7. Использование функции, возвращающей массив

```
Private sub Form_Load()
    Dim b As Byte, I As Integer
    Dim ReturnArray() As Byte
    ReturnArray() =  ArrayFunction(b)
    For I=0 To Ubound(ReturnArray)
            Debug.Print ReturnArray(i) ' вывод массива в окно Immadiate
                                       ' (окно непосредственного наблюдения)
    Next I
End Sub
-------------------------------------------------------
Public Function  ArrayFunction(b As Byte) As Byte
    Dim x(2) As Byte
    b=Cbyte(10) ' функция Cbyte(N) преобразует число
                ' в переменную типа Byte
    x(0)=b
    x(1)=b+Cbyte(200)
    x(2)=b+b
    ArrayFunction = x
End Function
```

В данном примере в процедуре Form_Load массиву ReturnArray присваивается массив, возвращаемый функцией ArrayFunction, состоящий из трех элементов.

Передача параметров функциям и процедурам

При вызове функций вы уже обратили, видимо, внимание, что все функции при своей работе используют параметры или аргументы.

Параметры могут быть обязательными и необязательными. Необязательные параметры должны располагаться в конце списка параметров. Необязательные параметры объявляются опцией ***Optional***. Например, заголовок процедуры учета пользователей программы может выглядеть следующим образом:

```
Public Function CreateUser(UserId As String, _
    Password As String, Optional Description As String)
```

В данном примере последний параметр – описание является необязательным.

При передаче параметров функциям и процедурам используются также опции ***ByRef, ByVal и ParamArray***.

Опция ***ByRef*** – позволяет передавать параметры по ссылке. Это значит, что процедура или функция получают адрес переменной в памяти компьютера и значение переменной может изменяться. Данная опция используется по умолчанию.

Опция ***ByVal*** – позволяет передавать параметры по значению. В этом случае Visual Basic передает в функцию копию текущего содержания переменной. Значение переменной в памяти компьютера не изменяется.

```
Private Function FunctionName(ByVal  Parametr As  String) As Boolean
```

Опция ***ParamArray*** позволяет передать в функцию массив. Массив при этом должен быть объявлен как массив типа Variant:

```
Private Function FunctionName(ParamArray  Massiv()  As  Variant) As Boolean
```

Вызов функции может при этом иметь следующий вид:

```
Dim CodName As Boolean
-------------------------
CodName = FunctionName("Иван", "Петр", "Станислав"," Антонина")
CodName = FunctionName( "Кузьма"," Анна")
CodName = FunctionName("Данила", "Петр"," Ирина")
```

Из приведенных примеров видно, что функцию можно использовать с различным числом параметров.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие категории встроенных функций используются в VB?
2. Запишите основные арифметические, тригонометрические и обратные тригонометрические функции.
3. Какие функции применяются для управления выводом чисел?
4. Приведите синтаксис пользовательской функции.
5. Какими способами передаются параметры процедурам и функциям?
6. Как используются пользовательские функции?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с функциями языка програмирования Visual Basic.

Различают функции встроенные и внешние (пользовательские). Пользовательские функции используются так же, как и встроенные функции. Синтаксис пользовательских функций подобен синтаксису пользовательских процедур. Только при объявлении функций используется ключевое слово Function, а не Sub. Кроме того, результаты вычисления должны быть присвоены переменной, имя которой равно имени функции.

Параметры процедурам и функциям могут передаваться по ссылке (опция ByRef) или по значению (опция ByVal). В функции могут передаваться массивы опцией ParamArray.

4.4.4. ОПЕРАТОРЫ ДЛЯ УПРАВЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ

Операторы управления вычислительным процессом Visual Basic приведены в разделе 3. Это условный оператор If/Then/Else, оператор выбора Select Case, операторы циклов For/Next, While/Wend, Do/Loop.

Кроме того в Visual Basic для работы с перечисляемыми переменными используется оператор цикла ***For/ Each***. Синтаксис оператора:

```
For Each  <переменная цикла>  In объект
    [операторы]
Next  переменная цикла.
```

Пример 4.8:

```
Dim  nNumber As Integer
Dim  MyArray As Variant
MyArray = Array (3,6,9,9,5,2,3,1)
For Each nNumber In MyArray
    If  nNumber > 5 Then Debug.Print nNumber
Next nNumber
```

Здесь MyArray – массив, nNumber – текущий индекс элемента массива. Пока nNumber принадлежит массиву, цикл выполняется. Пользователю не надо заботиться о контроле числа элементов в массиве. Обязательное требование – переменная цикла должна быть переменной целого типа.

Пример 4.9 Процедура очистки объектов TextBox

```
Option Explicit
Dim ctl As Control
Private Sub cmdTextBoxClear_Click()
    For Each ctl In Controls
        If TypeOf ctl Is TextBox Then
            ctl.Text = ""
        End If
    Next ctl
End Sub
```

Процедура предназначена для очистки элементов управления TextBox.

Объявлена переменная ctl типа Control.

Программа проверяет, все объекты, установленные на форме и, если текущий объект является текстовым полем, очищает его.

В данном примере переменная ***ctl*** является переменной типа объект. В примере используется также функция ***TypeOf***, которая проверяет принадлежность текущего объекта ctl заданному типу объекта:

```
If TypeOf ctl Is TextBox
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие операторы используются в VB для управления вычислительным процессом?
2. Приведите синтаксис условных операторов.
3. Приведите синтаксис операторов цикла For/Next и For/Each.
4. Приведите синтаксис и схему алгоритма оператора While/Wend.
5. Приведите синтаксис оператора Do/Loop.
6. Приведите синтаксис и поясните принцип работы оператора Select Case.

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с операторами для управления вычислительным процессом:

- с условными операторами If/Then/Else, Select Case;
- с операторами цикла For/Next, For/Each, While/Wend, Do/Loop;
- с оператором цикла For/Each. Оператор For/Each используется для работы только с перечисляемыми переменными, поэтому переменная цикла должна быть переменной целого типа;
- с функциями форматирования данных Round и Format, которые используются для управления выводом информации в объекты или на печать. В функция Format используются стандартные и пользовательские шаблоны. Для управления выводом информации могут использоваться также дополнительные функции форматирования;
- с процедурами и функциями пользователя. Процедры и функции пользователя имеют по существу одинаковый синтаксис. Процедуры пользователя вызываются оператором Call или по имени. Функции пользователя используются так же как и встроенные функции языка Visual Basic.

4.4.5. МАССИВЫ

Классификация массивов

Понятие о массивах переменных было рассмотрено нами в разделе 3.3.4.

Чтобы лучше представлять, о чем пойдет речь, приведем вначале классификацию массивов.

Массивы в VB можно классифицировать по следующим признакам (рис.4.20.): числу

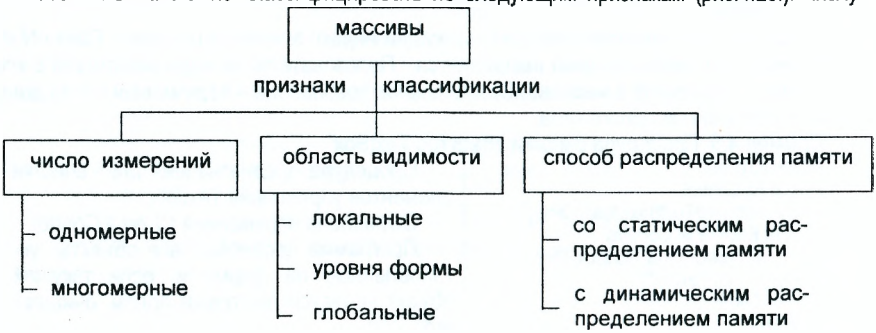


Рис.4.20. Классификация массивов

измерений, области видимости, способу распределения памяти. VB позволяет создавать одномерные и многомерные массивы. Число размерностей массива может достигать 60.

Нумерация элементов массива начинается с нуля. Для изменения индексации с нуля на единицу используется оператор ***Option Base N***, где N может принимать значения 0 или 1. Оператор Option Base записывается в раздел Главная контейнера (формы, модуля, класса). Однако при объявлении массивов можно задавать произвольные значения верхних и нижних границ массива.

Объявление массивов. Область видимости

Так же как и переменные, массивы могут быть локальными, уровня формы (модуля) и глобальными.

Локальные массивы видимы (доступны) только в данной процедуре. При выходе из процедуры данные теряются. Такие массивы объявляются оператором ***Dim*** или ***ReDim***: Dim Massiv1(10, 20). Локальные массивы модуля объявляются оператором ***Private***.

Массивы уровня формы доступны всем процедурам формы. Они объявляются оператором ***Dim*** в разделе Главная.

Глобальные массивы доступны всем процедурам проекта и объявляются оператором ***Public*** или ***Global*** в разделе Главная модуля (табл. 4.10).

Таблица 4.10 Операторы, используемые для объявления массивов

Локальные массивы		*Массивы уровня формы*	*Глобальные массивы*
процедуры	*модуля*		*объявляются в модуле*
Dim, ReDim	Private	Dim	Public, Global

Синтаксис объявления массивов:

<область видимости> <имя массива> (размерность массива) As <тип массива>

<область видимости> <имя массива> (нижняя_граница TO верхняя_граница) As <тип массива>

Примеры объявления массивов:

Dim A(10) As String – одномерный массив, содержит 11 элементов;

Dim B(5 TO 10, 1 TO 20) As Integer – двухмерный массив, имеет 6 строк и 20 столбцов. Нумерация строк начинается с 5, а нумерация столбцов с единицы.

Способы распределения памяти

Способ объявления массивов зависит от способа распределения памяти.

Массивы со статическим распределением памяти (для краткости – статические массивы) объявляются операторами ***Dim, Private, Public, Static:***

Static A (1 TO 5, 1 TO 2).

Эти массивы объявляются один раз и не меняют своих размерностей в процессе выполнения программы. Границы размерностей задаются только числами, ***нельзя*** использовать для указания размерности переменные.

Массивы с динамическим распределением памяти (для краткости – динамические массивы) объявляются в два этапа. Сначала они объявляются на уровне контейнера без указания размерности:

Dim AMassiv ( ) As Variant.

Затем с помощью оператора ***ReDim*** устанавливаются фактические размерности массива. В отличие от оператора Dim оператор ReDim используется только в процедурах. Оператор ReDim допускает использование переменных для указания размерностей массивов:

ReDim MassivA(5, 10) As Integer

ReDim MassivB(m, n) As Single

При использовании оператора ReDim нельзя изменять тип данных массива, кроме случая, когда тип массива объявлен как Variant.

Когда переопределяются размерности массива, есть опасность потерять содержимое. Чтобы при переопределении размерности массива не потерять данные, совместно с оператором ReDim используется опция ***Preserve:***

```
ReDim Preserve MassivA(10, 50)
```

Однако, если используется ключевое слово *Preserve*, то для многомерных массивов можно изменять только последнее измерение, а в последнем измерении можно менять только верхнюю границу индекса, например:

```
ReDim aArray(10,10) ' объявленный массив
ReDim Preserve aArray(10,20)        ' допускается
ReDim Preserve aArray(15,10)        ' не допускается
ReDim aArray(5 To 10,10 To 20) 'объявленный массив
ReDim Preserve aArray(5 To 10,10 To 25)     'допускается
ReDim Preserve aArray(5 To 10,15 To 20)     ' не допускается
```

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

Visual Basic имеет несколько функций для работы с массивами:

Array – создание массива типа Variant.

Lbound(ИмяМассива, Индекс) – возвращает нижнюю границу диапазона индекса массива. Индекс указывается только для многомерных массивов и определяет, к какому измерению массива применяется функция. Например:

```
Dim  aArray(5 To 10,15 To 20)
...
Lbound (aArray, 2) –  определяется нижняя граница второго измерения массива;
```

Ubound (ИмяМассива, Индекс) – возвращает верхнюю границу диапазона индекса массива.

Функции Lbound и Ubound имеют важное значение. При передаче в процедуру массивов их границы неизвестны, здесь на помощь приходят данные функции. Эти функции полезны также при присвоении данных одного массива другому. С помощью функций Lbound и Ubound можно определить размерность нового массива.

Например, требуется передать данные из двухмерного массива A в массив B:

```
' объявление масива B
 ReDim B(Lbound(A,1) To Ubound(A,1), Lbound(A,2) To Ubound(A,2) As Variant
 For i=Lbound(a,1) To Ubound(a,1)
    For j=Lbound(a,2) To Ubound(a,2)
        B(i,j)=A(i,j)
          Next j
       Next i
```

IsArray (ИмяПеременной) – проверка, является ли переменная массивом.

Пример 4.10. Ввод и вывод одномерного массива.

```
Option Explicit
Dim Massiv() As Variant
---------------------------------------------------------------
Sub VvodMassiv1(a() As Variant)
   Dim i As Integer, Ngraniza As Integer, Vgraniza As Integer
   Ngraniza = Val(InputBox("Укажите нижнюю границу массива", _
```

```
            "Ввод данных в массив"))
    Vgraniza = Val(InputBox("Укажите верхнюю границу массива", _
            "Ввод данных в массив"))
    ReDim Massiv(Ngraniza To Vgraniza) As Variant
    For i = Ngraniza To Vgraniza
        Massiv(i) = Val(InputBox("Введите" & Str(i) & " элемент массива"))
        Print "Massiv("; Str(i); ")="; Massiv(i) ' печать элемента массива
    Next i
End Sub
-------------------------------------------------------------------
Private Sub Form_Click()
    ' использование подпрограммы вода данных в одномерный массив
    Cls
    VvodMassiv1  Massiv
End Sub
```

В функции InputBox для отображения номера вводимого элемента массива формируется строка символов *"Введите" & Str$(i) & " элемент массива"*, в которой для объединения фрагментов строки используется символ "**&**" амперсенд (коммерческое И).

Пример 4.11. Присваивание значений элементов одного одномерного массива элементам другого массива.

```
Option Explicit
Dim Massiv() As Variant, NewMassiv() As Variant
-------------------------------------------------------
Sub CopyMassiv(A() As Variant, B() As Variant)
    Dim i As Integer
    ReDim B(LBound(A) To UBound(A))
    For i = LBound(A) To UBound(A)
        B(i) = a(i)
        Print "NewMassiv("; Str(i); ")="; B(i)
    Next i
End Sub
-------------------------------------------------------
Private Sub Form_Click()
    Cls
    VvodMassiv1 Massiv
    CopyMassiv Massiv, NewMassiv
End Sub
```

Visual Basic 6 позволяет непосредственно присваивать значения элементов одного массива элементам другого массива:

```
Sub CopyMassiv1(A() As Variant, B() As Variant)
    Dim i As Integer
    ReDim B(LBound(A) To UBound(A)) As Variant
    B = A ' присвоение значений элементов массива A элементам массива B
    For i = LBound(A) To UBound(A)
        Print "B(";str(i); ")=";B(i)
    Next i
End Sub
```

Однако при выполнении данной операции необходимо учитывать типы переменных и размерности массивов. Если новый массив динамический, то операция обмена всегда успешна. В этом случае число элементов массива в левой части оператора присваивания при необходимости изменяется.

Использование функции Array для объявления массива

Функция Array присваивает список значений переменной типа Variant. При этом переменная автоматически преобразуется в одномерный массив.

Пример 4.12. Использование функции Array

```
Option Explicit
Dim A As Variant, i As Integer
-----------------------------------------
Private Sub Form_Click()
        A = Array(10, 12, 53, 48, 54)
        For i = 0 To 4
                Print A(i)
         Next i
End Sub
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Приведите классификацию массивов.
2. Приведите примеры одномерного и многомерного массивов.
3. Какими операторами объявляются статические массивы?
4. Каков порядок объявления динамических массивов?
5. Какими операторами объявляются динамические массивы?
6. Приведите текст программы для ввода данных в одномерный массив.
7. Перечислите и приведите примеры использования функций для работы с массивами.
8. Напишите подпрограмму для обмена данными двух массивов.
9. Приведите текст программы для ввода данных в одномерный массив.
10. Напишите программу для сортировки одномерного массива.

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с особенностями использования массивов в Visual Basic. Изучили классификацию массивов и способы их объявления. Особенностью динамических массивов является то, что они объявляются дважды: первоначально в разделе общие формы (модуля) без указания размерности, а затем в процедуре с указанием размерности. Ввод и вывод данных в массивы осуществляется с помощью циклов. Важное значение при работе с массивами имеют функции определения границ массива Lbound и Ubound.

4.4.6. МАССИВЫ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ

Понятие о массиве элементов управления

Microsoft Visual Basic 6.0 предоставляет пользователям несколько дополнительных средств, помогающих создавать эффективные и гибкие приложения. Одно из них – массив элементов управления (control array). Массив элементов управления представляет собой группу элементов управления с одинаковыми именами, типом и обработчиками событий. Однако элементы такого массива сохраняют и индивидуальные значения свойств. Чтобы различить эти элементы управления используют такие их свойства, как *Index*, *Caption* или *Tag*, *TabIndex*.

Массив элементов управления имеет, по крайней мере, один элемент, у которого значение свойства Index равно нулю. Максимальное значение свойства Index равно 32767, если это не ограничено размером оперативной памяти компьютера.

Обычно такие массивы применяются для элементов управления "меню", групп переключателей или флажков, а также для вывода на экран значений элементов массивов.

У массивов элементов управления имеется три существенных преимущества:

- позволяют добавлять новые элементы управления в период выполнения программы. Это особенно важно, когда заранее не известно, сколько новых элементов понадобится в период выполнения. Кроме того, добавление элементов управления только по мере необходимости обеспечивает экономию системных ресурсов. Элементы управления, добавляемые в период выполнения, называются динамическими;
- каждый новый элемент, добавляемый в массив, наследует общие для массива процедуры обработки событий, что облегчает программирование;
- элементы управления в массиве способны разделять код. Например, если на форме имеется несколько текстовых полей, предназначенных для ввода

четных чисел, то можно использовать общий обработчик событий для проверки правильности вводимых чисел:

```
Private Sub Text1_Change(Index As Integer)
    If IsNumeric(Val(Text1(Index).Text)) And _
        Val(Text1(Index).Text) Mod 2 = 0 Then
        MsgBox "Ввод правильный"
        Exit Sub
    Else
        MsgBox "Ошибка ввода"
    End If
End Sub
```

Программа в обработчике события Change объекта Text1 проверяет, если значение свойства Text объекта Text1 числовое (проверяет функция IsNumeruc) и это значение четное, то происходит выход из процедуры, иначе выдается сообщение об ошибке.

Создание массивов элементов управления на этапе разработки

Имеется несколько способов создания массивов элементов управления на этапе разработки:

- копирование и вставка элемента управления;
- присвоение двум существующим элементам управления одного типа одинаковых имен;
- установка свойства Index в окне свойств.

Алгоритм работы по созданию массива элементов управления путем копирования:

1. Поместите на форму первый элемент управления и задайте начальные значения свойств, общие для всех элементов управления.
2. Установите свойство Name этого элемента.
3. Скопируйте элемент управления и вставьте его в форму. На запрос программы: "Создать массив элементов управления?" ответьте утвердительно. Запрос выдается только при вставке первой копии элемента управления.
4. Повторяйте операцию вставки, пока на форму не будет добавлено нужное количество элементов управления.

Алгоритм работы по созданию массива элементов управления путем присвоения двум однотипным элементам управления одинаковых имен:

1. Поместите на форму два одинаковых элемента управления.
2. Присвойте имя первому элементу управления.
3. Присвойте такое же имя второму элементу управления. Программа выдаст запрос о создании массива элементов управления. Ответьте утвердительно.

При добавлении на форму последующих элементов управления этого же типа запроса на создание массива управления выдаваться не будет.

Алгоритм создания массива элементов управления установкой значения индекса

1. Установите на форму элемент управления.
2. Присвойте свойству Index значение 0.
3. Для добавления в массив новых элементов управления на этапе разработки теперь достаточно присвоить новому элементу имя базового элемента. Значение индекса будет присвоено автоматически.

Предупреждение. Индекс первого элемента управления может быть отличным от нуля. Однако в этом случае при добавлении нового элемента управления его индекс может оказаться меньше, чем индекс существующих элементов управления, что создаст массу неудобств. Поэтому при создании нового массива элементов управления первый индекс всегда должен быть равен нулю.

При программировании массива элементов управления используется следующий синтаксис:

```
ИмяЭлементаУправления (Индекс).Свойство = <Значение>
```

Например:

```
txtText1(0).Text = "Фамилия"
txtText1(1).Text = "Имя"
```

Массив элементов управления приобретает некоторые новые свойства, по сравнению с одиночным элементом, например, свойство Count – количество элементов управления в массиве, в процедуре обработки события Change появляется в качестве аргумента системная переменная Index:

```
Private Sub Text1_Change(Index As Integer).
```

Динамическое добавление элементов управления в период выполнения

Для динамического добавления элементов управления на форму используется оператор ***Load*** и метод ***Add***.

Оператор Load позволяет только добавлять новые элементы к уже существующим элементам управления. Для добавления нового элемента управления введите команду

```
Load объект(индекс).
```

Оператор Load копирует значения всех свойств объекта из первого элемента массива, кроме *Visible*, *Index* и *TabIndex*,. Поэтому вновь созданный элемент управления размещается под исходным элементом управления и невидим. Чтобы сделать этот элемент управления доступным для управления, необходимо сместить его в сторону используя свойства *Top* и *Left*, а также присвоить свойству ***Visible*** значение ***True,*** например:

```
объект(индекс).Top = объект(индекс-1).Top + объект(0). Height
объект(индекс).Left = объект(индекс-1).Left + объект(0). Width
объект(индекс).Visible = True
```

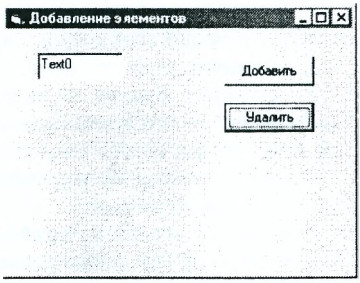


Рис.4.21. Форма для добавления и удаления элементов массива

Попытка загрузки уже существующего элемента управления вызывает ошибку периода выполнения. Если свойство Index у объекта не установлено, то также выдается сообщение об ошибке.

Удаление элементов массива элементов управления осуществляется оператором ***Unload***: Unload объект(Index)

Нельзя удалять исходный (нулевой) элемент массива, так как после этого его нельзя будет восстановить программным путем.

Пример 4.13. Добавление и удаление элементов управления

Поместим на форму (рис. 4.21) один элемент управления TextBox – Text1 и присвоим свойству Index этого элемента значение 0. Cdjqcnde Text присвоим значение "Text0". Поместим на форму также две кнопки для добавления и удаления элементов массива (cmdAdd, cmdDelete).

```
Private Sub cmdAdd_Click() ' процедура добавления элементов управления
    Dim i As Single
    i = Text1().Count   ' определяем номер очередного индекса
    Load Text1(i)       ' загружаем новый элемент _ управления на форму
    Text1(i).Text = "Текст" & Str$(i) ' присваиваем значение свойству Text
                                      ' нового элемента управления
    Text1(i).Top = Text1(i - 1).Top + Text1(0).Height
    ' смещаем новый элемент вниз на высоту элемента управления
    Text1(i).Visible = True     ' делаем видимым новый элемент управления
End Sub
-------------------------------------------------------------
Private Sub cmdDelete_Click() ' процедура удаления элементов управления
    Dim i As Single
    i = Text1().Count – 1
    ' определяем индекс последнего элемента управления
    ' индекс на единицу меньше, чем число элементов управления,
    ' так как нумерация начинается с нуля
    If i > 0 Then
       Unload Text1(i)
    End If
End Sub
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое массив элементов управления?
2. В чем заключается преимущество в использовании массивов элементов управления по сравнению с отдельными элементами управления?
3. Как создать массив элементов управления на этапе разработки программы?
4. Как добавить элементы управления во время выполнения программы?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы встретились с новым понятием "массив элементов управления". Массивы элементов управления являются средством, обеспечивающим экономию системных ресурсов и позволяющих сократить текст программы за счет использования общего обработчика событий. Используются для вывода информации, при создании меню пользователя, при управлении флажками и переключателями.

4.4.7. УПРАВЛЯЮЩИЙ ЭЛЕМЕНТ СЕТКА

Управляющий элемент ***MSFlexGrid*** – сетка предназначен для вывода данных на экран. Вводить данные в ячейки сетки непосредственно нельзя.

Сетки нет среди стандартных элементов панели ToolBox. Для ее загрузки необходимо ввести команду ***Project, Components*** и установить флажок для элемента управления ***Microsoft Flex Grid Control 5.0.***

Основные свойства сетки

Сетка имеет более 80 свойств, 20 событий и 10 методов.

Свойства сетки можно легко настраивать с помощью диалоговой панели (рис.4.22): щелкните по полю ***Custom*** окна ***Properties*** – появится кнопка *троеточие*, щелкните мышью по этой кнопке – появится диалоговая панель. В закладках диалоговой панели можно установить необходимые параметры (с назначением параметров, представленных на рисунке, познакомимся по мере изложения материала). Для настройки свойств сетки можно с успехом использовать также и окно свойств.

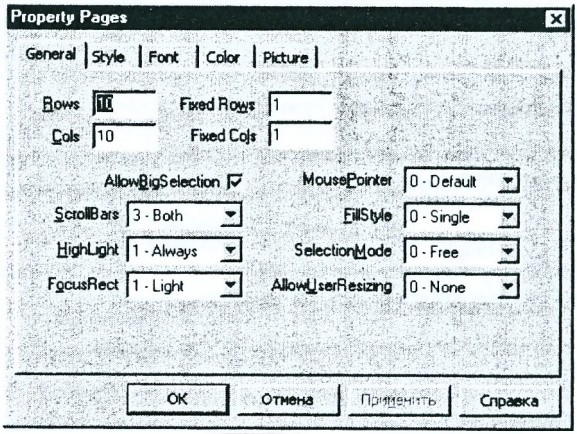


Рис. 4.22. Панель свойств сетки

Рассмотрим основные свойства сетки.

Top – расстояние сетки от верхнего края формы, ***Left*** – расстояние сетки от левого края формы; ***Heigt*** – высота; ***Width*** - ширина; ***Enabled*** – доступность; ***Visible*** – видимость.

Cols, Rows – устанавливают число столбцов и строк.

Col, Row - возвращают/ устанавливают номер столбца и строки.

Нумерация строк и столбцов начинается с нуля.

FixedCols, FixedRows, FixedAlignment – фиксация строк, столбцов или значений. Сетка используется, как правило, для размещения таблиц данных. Если размеры сетки на форме небольшие, а число строк и колонок в таблице значительное, то при перемещении по сетке начальные строки и колонки пропадают с экрана. Это создает затруднения при анализе и чтении данных. Чтобы избежать такого неудобства нужно зафиксировать шапку таблицы. Синтаксис использования свойств:

```
Имя Сетки. FixedRows = ЧислоФиксированныхСтрок%
    Grid1. FixedRows = 2
    Grid1. FixedCols = 1
```

Зафиксированы две строки и один столбец. "ЧислоФиксированныхСтрок" может быть только константой или переменной целого типа.

ScrollBar - линейка прокрутки, имеет 4 значения: 0- выводится автоматически, 1- горизонтальная, 2- вертикальная, 3 – обе.

ColPosition, RowPosition – позволяют перемещать целые колонки и столбцы по сетке. Синтаксис использования свойства:

```
ИмяСетки.ColPosition(N)=значение%
ИмяСетки.RowPosition(N)=значение%
```

здесь N номер колонки или строки.

ColWidth, RowHeight - ширина и высота столбца. Синтаксис использования этих свойств аналогичен предыдущему примеру:

```
ИмяСетки.ColWidth = значение%
ИмяСетки.RowHeight = значение%
```

Text, TextMatrix - возвращает или устанавливает текст, хранящийся в текущей ячейке.

Свойство Text сетки аналогично свойству Text объекта TextBox. При использовании свойства Text для ввода данных в ячейку необходимо сначала активизировать эту ячейку (или иначе - переместить фокус на эту ячейку), присвоив свойствам Row и Col нужные значения.

Пример 4.14. Необходимо ввести данные в ячейку на пересечении пятой строки и четвертого столбца. Для реализации этой задачи необходимо написать следующий фрагмент программы:

```
Grid1.Row = 4
Grid1.Col = 3
Grid1.Text = "Брестская крепость - герой"
```

Свойство TextMatrix имеет синтаксис:

```
TextMatrix (номер строки, номер столбца) = строка
Grid1.TextMatrix(i,j)= "2003"
```

Это свойство позволяет считывать или вносить текст в произвольную ячейку без изменения значений свойств Row и Col.

ColAlignment - выравнивание текста в ячейках. Можно использовать 10 возможных значений свойства Alignment для управления выравниванием информации в ячейках:

Значение	*Выравнивание*	
	по горизонтали	*по вертикали*
0	по левому краю	по верхнему краю
1	по левому краю	по центру
2	по левому краю	по нижнему краю
3	по центру	по верхнему краю
4	по центру	по центру
5	по центру	по нижнему краю
6	по правому краю	по верхнему краю
7	по правому краю	по центру
8	по правому краю	по верхнему краю
9	по левому краю	по центру

Синтаксис: ИмяСетки. ColAlignment (индекс) = Значение%

```
Grid1.ColAlignment(i) = 2
```

LeftCol, TopRow – номер самого левого столбца и самой верхней строки, которые будут отображаться в сетке. Эти свойства используют, когда таблица не помещается в форме:

```
Имя Сетки. LeftCol = ЛеваяКолонка%
Имя Сетки. TopRow = ПерваяСтрока%
```

GridsLines – контролирует отображение разделительных линий.

ScrollBars – контролирует отображение линеек прокрутки.

Пример 4.15. Ввести в ячейку на пересечении 4-й строки и 5-го столбца текст "Писатель". Текст выровнять по центру. Очистить первую ячейку.

Решение: Выделить ячейку на пересечении строки с номером 3, и столбца с номером 4, записать в нее текст. Выровнять текст в ячейке посередине. Очистить ячейку в нулевой строке и нулевой колонке:

```
Private Sub Form_Click()
   ' Ввод данных
   MSFlexGrid1.TextMatrix(3, 4) = "Писатель"
   MSFlexGrid1.ColAlignment(3) = 3
   ' Очистка ячейки
   MSFlexGrid1.TextmMatrix(0,0) = ""
End Sub
```

Другой вариант:

```
Private Sub Form_Click()
        ' Ввод данных
        MSFlexGrid1.Row = 3     ' устанавливаем курсор в ячейку на
        MSFlexGrid1.Col = 4     ' пересечении 4 строки и 5 столбца
        MSFlexGrid1.Text = "Писатель"
        MSFlexGrid1.ColAlignment(3) = 3
End Sub
Private Sub Form_DblClick()
    ' Очистка ячейки
    MSFlexGrid1.Col = 1
    MSFlexGrid1.Row = 1
    MSFlexGrid1.Text = ""
End Sub
```

Свойства для выделения ячеек внутри таблицы

ColSel, RowSel – выделение ячеек.

Сначала устанавливается начальная ячейка, а затем указывается область:

```
MSFlexGrid1.Row=0: MSFlexGrid1.Col=0
MSFlexGrid1.ColSel=4: MSFlexGrid1.RowSel=3
```

Выделяются колонки с 0 до 4 и строки с 0 до 3

Clip - используется для считывания и установки содержимого выделенной части таблицы:

```
Имя сетки.Clip=Строка
```

В строке содержатся данные, предназначенные для разных столбцов. Эти данные должны отделяться друг от друга символами табуляции, а строки должны отделяться символами возврата каретки.

Пример 4.16. Выделить область, включающую две строки и три столбца. Начальная ячейка 1,1. В ячейки поместите числа 1, 2 , 3, 4, 5, 6.

```
Private Sub Form_Click()
MSFlexGrid1.Col = 1:  MSFlexGrid1.Row = 1
MSFlexGrid1.ColSel = 3: MSFlexGrid1.RowSel = 2
S$ = Str$(1) + vbTab + Str$(2) + vbTab + Str$(3) + vbCr _
  + Str$(4) + vbTab + Str$(5) + vbTab + Str$(6)
MSFlexGrid1.Clip = S$
End Sub
```

Здесь *vbTab* – константа табуляции;

VbCr - константа возврат каретки.

FillStyle - автозаполнение. Заполнение выделенной области данными, внесенными в одну ячейку. Может принимать два значения: 0 и 1. 0 – одно значение, 1 - повторять. По умолчанию значение равно 0.

HighLight – сообщение о выделении ячейки: 0 - никогда, 1 – подсвечивание ячейки; 2 – ячейка подсвечена даже если потеряла фокус.

AllowBigSelection – разрешает или запрещает выделять столбцы щелчком мыши по заголовку. Имеет два значения: True и False.

AllowUserResizing – устанавливает возможность изменения размеров строк и столбцов с помощью мыши: 0 - нельзя, 1 – можно менять размеры колонок; 2 – можно менять размеры строк; 3 – можно менять и то и другое.

Sort – сортировка сетки: 0 – нет сортировки; 1 – по возрастанию; 2 – по убыванию ; 3 – по возрастанию, но не конвертировать строки в числа; 4 – по убыванию, но не конвертировать строки в числа; 5 – по возрастанию без учета регистра; 6 – по убыванию без учета регистра; 7 – по возрастанию с учетом регистра; 8 – по убыванию с учетом регистра; 9 – для сравнения используется событие ***Compare***.

События и методы сетки

EnterCell, LeaveCell – выделение и отмена выделения ячейки при щелчке мыши.

RowColChange - сохранение информации из глобальной переменной в текущей ячейке.

AddItem – вставка строк.

MsFlexGrid1.AddItem Item$[,номер строки]

Строка в переменной *Item$* помещается в первую колонку новой записи. Дополнительный параметр *номер строки* дает возможность указывать, перед какой строкой следует добавить новую строку. По умолчанию строка вставляется в конце таблицы.

RemoveItem – удаление строки.

MsFlexGrid1. RemoveItem <номер строки>

Пример 4.17. Использование свойств и методов сетки (рис.4.23).

Программа предназначена для демонстрации ввода данных в выделенную ячейку, вставки новых строк выше выделенной строки и удаления выделенных строк.

1. Добавьте на панель инструментов сетку ***MSFlexGrid*** командой ***Project, Components,*** установите флажок для элемента управления ***Microsoft Flex Grid Control 5.0.***
2. Установите на форму элементы управления: Сетку – имя Grid1, число строк –15,

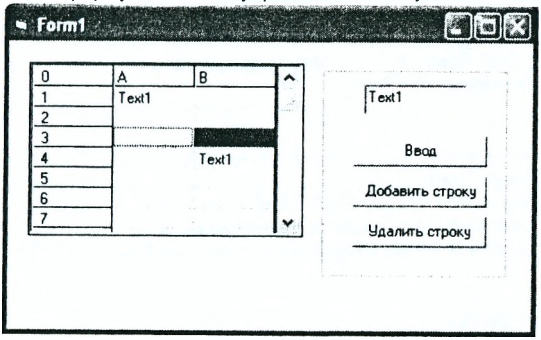


Рис. 4.23. Сетка MSFlexGrid

число столбцов 10, по умолчанию зафиксированы один столбец и одна строка; Окно ввода – Text1 и командные кнопки: Ввод - cmdVvod, Добавить строку - cmdInsert, Удалить строку - cmdDelete.

Окно ввода служит для ввода данных в сетку. Так как вводить данные в сетку непосредственно нельзя, то для этой цели используется окно ввода: данные вводятся в окно ввода, а из окна ввода выводятся на сетку. Кнопка Ввод служит для выполнения операции ввода данных в сетку. Ниже приведен фрагмент программы для управления вводом данных в сетку.

```
Option Explicit
Dim i As Integer, j As Integer, S As String
Dim Ab As String, Msg As String
```

```
Private Sub Form_Load()
    ' начальная настройка параметров сетки: нумерация  строк и столбцов
     Ab = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    ' Ab – символьная переменная, содержит символы латинского алфавита
    For i = 0 To Grid1.Rows – 1        ' нумерация строк
        Grid1.TextMatrix(i, 0) = Str(i)
    Next i
     For j = 1 To Grid1.Cols – 1        ' нумерация столбцов
        If Grid1.Cols < 26 Then
            Grid1.TextMatrix(0, j) = Mid$(Ab, j, 1)
             ' из строки Ab  извлекается текущий символ и
             ' присваивается заголовку столбца
        End If
     Next j
End Sub
---------------------------------------------------------------
Private Sub cmdDelete_Click()
     ' процедура удаления выделенной строки
     Grid1.RemoveItem Grid1.RowSel
     For i = 0 To Grid1.Rows – 1                ' перенумерация строк после
        Grid1.TextMatrix(i, 0) = Str(i)    ' удаления выделенной строки
     Next i
End Sub
---------------------------------------------------------------
Private Sub cmdInsert_Click()
' процедура вставки строки выше строки выделения.
     Msg ="Вставка строки"
     Grid1.AddItem Msg, Grid1.RowSel
     For i = 0 To Grid1.Rows – 1          ' пернумерация строк
        Grid1.TextMatrix(i, 0) = Str(i)
     Next i
End Sub
---------------------------------------------------------------
Private Sub cmdVvod_Click()
     Grid1.TextMatrix(Grid1.RowSel, Grid1.ColSel) = Text1.Text
     ' Текст из Окна ввода Text1 вводится в активную ячейку.
     ' Адрес выделенной  ячейка возвращается свойствами RowSel и ColSel
End Sub
---------------------------------------------------------------
Private Sub Form_DblClick()
     Grid1.TextMatrix(Grid1.RowSel, Grid1.ColSel) = ""
     ' Очистка выделенной  ячейки
End Sub
Private Sub Grid1_LeaveCell()
     Grid1.CellBackColor = QBColor(15)
     ' закраска ячейки белым цветом при отмене выделения ячейки.
 End Sub
Private Sub Grid1_LostFocus()
     Grid1.CellBackColor = QBColor(15)
     '  закраска ячейки белым цветом при потере фокуса сеткой
End Sub
Private Sub Grid1_SelChange()
  Grid1.CellBackColor = QBColor(14)
    ' закраска ячейки желтым цветом при выделении
End Sub
```

Из приведенного фрагмента программы можно сделать вывод, что возможности по управлению сеткой достаточно богаты, но все надо программировать вручную. Не следует, однако, пытаться превратить вашу программу в некое подобие электронной таблицы Excel. При необходимости проанализировать данные, рассчитанные в программе, можно передать их в электронную таблицу и там их проанализировать средствами Excel (см. раздел 4.11).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначен элемент MSFlexGrid?
2. Как поместить элемент MSFlexGrid на панель элементов управления ToolBox?
3. Как установить число строк и столбцов на сетке?
4. Как вызвать окно диалога для настройки параметров сетки?
5. Напишите фрагмент программы для присвоения значения ячейке на пересечении пятой строки и третьей колонки.
6. Как зафиксировать первую строку?
7. Напишите фрагмент программы для установки начальных параметров сетки.
6. Напишите фрагмент программы для добавления строки.

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с новым элементом управления – сеткой MSFlexGrid. Данный элемент управления является удобным средством для представления результатов вычисления в табличной форме. При работе с массивами очень удобным для использования является свойство сетки TextMatrix. Это свойство позволяет непосредственно передавать данные из двухмерного массива в сетку, используя переменные цикла. Свойство Text сетки можно использовать, когда курсор ввода перемещается в заданную ячейку.

4.5. РАЗРАБОТКА ИНТЕРФЕЙСА ПРИКЛАДНЫХ ПРОГРАММ

4.5.1. ПРИНЦИПЫ РАЗРАБОТКИ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Интерфейс пользователя – связующее звено между пользователями и функциональностью приложения.

Основные пользователи приложения называются целевой аудиторией. Зная их потребности, можно достаточно легко создать пользовательский интерфейс. Хорошо продуманный интерфейс упрощает освоение приложения и работу с ним.

Базовые принципы дизайна такие, как композиция и цвет, применимы и к изображению на мониторе компьютера. Для создания эффективного интерфейса, как говорят специалисты, не надо быть художником: главное соблюдать базовые принципы. И тогда интерфейсом будет легко пользоваться. От внешнего вида интерфейса и заложенных в него концепций прямо зависит и то, какой будет поддерживающая его программа.

Основным элементом пользовательского интерфейса любого приложения, разрабатываемого на Visual Basic является форма. На нее добавляются элементы управления и меню, которые обеспечивают доступ к функциональности приложения.

К базовым принципам дизайна относятся: *композиция; цвет; изображения и значки; шрифт; меню.*

Композиция – размещение элементов интерфейса. Размещение элементов на экране должно быть удобным и приятным на глаз, создавать максимальные удобства в использовании приложения. Композиция должна учитывать такие факторы, как *простота, разметка элементов, единообразие, узнаваемость, легкость восприятия.*

Простота – интерфейс не должен быть тяжеловесным. Он не должен копировать реальный объект. Необходимо использовать такие элементы, как поля ввода, списки, флажки, переключатели, предлагать значения отдельных полей по умолчанию, группировать поля по функциональному признаку.

Разметка – часто используемые элементы должны бросаться в глаза, находясь в самых выгодных позициях; менее значимые элементы можно сделать менее заметными; самый важный элемент должен находиться в левом верхнем углу экрана.

Кнопки OK или Следующий обычно размещаются в нижней правой части экрана. В окнах диалога кнопки располагаются справа или внизу формы.

Необходимо также логически группировать информацию по назначению или взаимосвязанности. Например, поля для имен и адресов должны располагают рядом. Для логического группирования элементов во многих случаях удобно пользоваться рамкой.

Единообразие или ***согласованность.*** Во всем приложении должен быть единый стиль. Рекомендуется придерживаться стиля в существующих клиентских приложениях вроде Microsoft Word, Microsoft Excel. Следует ограничиваться в выборе элементов управления, стараться использовать их по назначению. Например, кнопку управления можно сделать с помощью любого элемента управления, имеющего события Click и Double Click. В то же время имеется специальный элемент управления Command Button, который имеет специфические свойства, присущие только данному элементу управления.

Узнаваемость - определяется визуальными элементами, подсказывающими назначение компонентов пользовательского интерфейса. Например, поля ввода имеют рамку и белый фон, кнопки имеют трехмерное оформление.

Легкость восприятия. Элементы пользовательского интерфейса должны быть отделены достаточным пространством, чтобы они не выглядели слишком нагроможденными друг на друга и их можно было легко воспринимать. Если на форме слишком много элементов, то найти нужные будет непросто. Выравнивание элементов по горизонтали и вертикали тоже улучшает восприятие. Этой цели служат команды *Align* - выравнивание, *Make Same Size* – установка одинакового размера, *Horizontal Spacing* – горизонтальные промежутки, *Vertical Spacing* – вертикальные промежутки, *Center in Form* – центрирование меню *Format*.

Цвет - цвет оживляет интерфейс, но только если используется в меру. Важные участки пользовательского интерфейса можно выделить контрастным цветом. Не рекомендуется сочетание таких цветов, как красный и зеленый, так как некоторые люди, страдающие дальтонизмом, не смогут прочесть красный цвет на зеленом фоне.

Количество используемых цветов лучше ограничить и придерживаться во всем приложении определенной цветовой схемы.

Изображения и значки – картинки и значки также оживляют приложение, но так же как и в случае применения других интерфейсных элементов, главное – целесообразность и чувство меры. Изображения могут передавать информацию без текста, но разные люди воспринимают это поразному. Значки, помещаемые на панель инструментов, могут давать представление о скрытой за ними функциональности. Целесообразно использовать известные, ставшие привычными значки, например, такие, как на рис. 4.24: корзина - папка для удаленных файлов; часы – напоминание о времени; желтый прямоугольник - папка, диск с дисководом – устройство для чтения компакт дисков; принтер – печать документов и т. д.

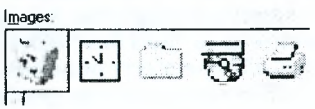


Рис.4.24. Значки

Шрифты – несут важную информацию пользователю. Некоторые шрифты легко читаются при разных разрешениях экрана и на разных типах мониторов. Выбирайте один или два простых шрифта. Рекомендуются Arial или Times New Roman. Декоративные шрифты хорошо выглядят обычно только на бумаге.

Меню – меню и панели инструментов позволяют структурировать доступ к командам и инструментам. Должное планирование и правильный дизайн меню и панелей инструментов помогут пользователям быстрее понять назначение и возможности Вашего приложения. Если меню хорошо продумано, то уже одно знакомство с ним позволит пользователю составить представление о его возможностях.

Рекомендуется придерживаться стиля, принятого сейчас в клиентских приложениях типа Microsoft Word, Microsoft Excel и др. Рекомендуется также контролировать структуру меню и редактировать ее в зависимости от контекста работы приложения, динамически добавляя или удаляя пункты меню, включая или отключая какие-либо команды. Многие пользователи привыкли пользоваться контекстными меню, поэтому такие меню необходимо предусмотреть в разрабатываемом приложении.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите базовые принципы разработки пользовательского интерфейса.
2. Что входит в понятие композиция при разработке интерфейса?
3. Назовите основные принципы композиции и дайте им краткую характеристику.
4. Назовите основные компоненты пользовательского интерфейса.

ЗАКЛЮЧЕНИЕ

В данном разделе мы познакомились с основными принципами разработки интерфейса прикладных программ. Главное – соблюдение чувства меры, удобство размещения элементов управления и узнаваемость.

4.5.2. ФОРМА И ЕЕ СВОЙСТВА

Форма (рис. 4.25.) - это средство общения программы с "внешним миром", т.е. с пользователем. Форма выполняет роль контейнера. Это значит, что в нее можно помещать другие объекты. Свойствами контейнера обладают и такие элементы управления, как Frame – рамка, PictureBox - картинка и ToolBar – панель инструментов.

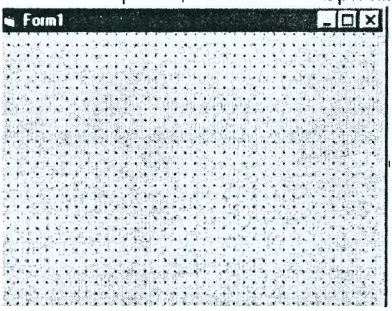


Рис. 4.25 Форма

Форма имеет все элементы стандартного окна Windows: строку заголовка, в которой указано наименование формы; кнопку системного меню – в левой части строки заголовка; кнопки свертывания, развертывания и закрытия окна – в правой части строки заголовка.

Кнопка свертывания сворачивает форму в значок (режим Minimize) и помещает его в нижней части экрана. Кнопка развертывания разворачивает окно на весь экран (режим Maximize). После развертывания окна кнопка развертывания заменяется кнопкой восстановления первоначального состояния окна. Кнопка закрытия закрывает окно.

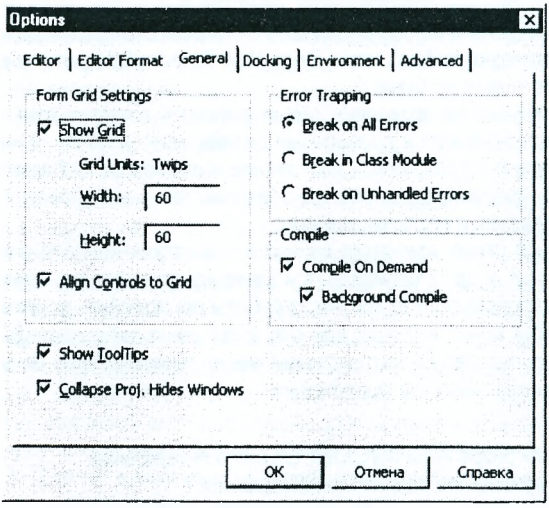


Рис.4. 26. Окно диалога настройки параметров сетки

Для удобства размещения элементов в форме, на нее можно установить сетку. По умолчанию, сетка выведена на форму. При необходимости, сетку можно удалить или изменить расстояние между ячейками. Для вывода сетки на форму используется команда ***Tools,Options*** (или комбинация клавиш [***Alt+T,O***]). После появления окна диалога *Options* (рис. 4.26) выберите закладку *General* и установите флажки *Show Grid* – показать сетку и *Align Controls to Grid* – "привязать" объекты к сетке. Привязку элементов управления к сетке можно выполнить и командой ***Format,Align,To Grid***. Расстояние между соседними ячейками устанавливается в строках ввода опции *Grid Units: Widht* –ширина и *Height* – высота. Расстояние измеряется в твипах.

Каждая форма сохраняется в проекте в виде отдельного файла с расширением имени файла ***FRM***. Этот файл содержит описание рабочей среды и тексты программ, относящиеся к элементам управления и форме. Формы сохраняются как обычные текстовые файлы.

Формы могут быть нескольких видов: *обычные* формы, *модальные* формы и *MDI* – формы, *дочерние* и *диалоговые*.

Модальность означает, что выполнение приложения возможно только после закрытия окна формы. Установка модальности осуществляется при загрузке формы и выполняется командой

ИмяФормы.Show vbModal или ИмяФормы.Show 1

Здесь vbModal – константа Visual Basic, которая имеет значение 1.

MDI - формы служат для организации совместной работы нескольких форм, которые называются дочерними.

Диалоговые формы служат для организации взаимодействия пользователя с программой. Диалоговые формы, как правило, являются модальными.

Свойства формы

Основные свойства формы приведены в табл. 4.11.

Таблица 4.11 Основные свойства формы

Свойства	*Значение по умолчанию*	*Комментарий*
Name	"Form 1"	Имя формы. Присваивается при разработке. Префикс *frm* (например, *frmProject1*)

Продолжение таблицы 4.11

Apperance	1	Внешний вид формы: 0 – плоская, 1 – объёмная.
BorderStyle	2	Внешний вид принимаемый по умолчанию имеется возможность изменения размеров формы при помощи мыши (рис. 4.25).
		0 - без рамки. Нет кнопок и заголовка. Используется для экранов с заставками. Изменять размеры и перемещать нельзя.
		1 - нельзя изменять размеры. Возможны операции Minimize и Maximize. Имеется две кнопки: кнопка системного меню и кнопка закрытия окна.
		3 - толстая рамка, размеры которой менять нельзя. Используется для создания диалоговых панелей. Имеется две кнопки: кнопка системного меню и кнопка закрытия окна.
		4 – нельзя изменять размеры. Используется для вывода окна с кнопкой Close. Имеется одна кнопка закрытия окна.
		5 - то же, что и 4, но можно изменять размеры окна.
Caption	"Form 1"	Текст заголовка. Устанавливается при разработке. Можно также изменять программным путем.
ControlBox	1	1 - есть кнопка системного меню. 0 - нет кнопки системного меню (не рекомендуется убирать кнопку системного меню).
Enabled	True	Доступность формы. Если значение свойства установлено True, то форма реагирует на события, False – форма не реагирует на события.
Font	MS Sans Serif	Возможна настройка параметров шрифта с помощью окна диалога: тип, стиль, размер, эффекты, размещение. При щелчке мышью по свойству Font в строке ввода появляется значок эллипсис - (или троеточие). Если щелкнуть мышью по этому [•••] значку, то открывается окно диалога для настройки шрифтов.
Height Width	2880 3840	Высота и ширина формы в твипах. В одном сантиметре 567 твипов.
Icon	Согласно стандартным настройкам Windows	Определяет значок, выводимый при минимизации программы на линейку инструментов или на рабочий стол в случае обычного исполняемого файла Windows. Форму значка можно изменить, загрузив новый файл с помощью окна диалога. Предварительно необходимо найти на компьютере папку с файлами, имеющими расширение *.ICO*.
Left Top	0 0	Определяют положение формы: расстояние от левого края экрана до формы, и от верхнего края экрана до формы, соответственно. Другой способ установки положения формы на экране состоит в использовании окна Form Layout. Перетащите мышью значок формы в нужное положение. Это окно работает только после запуска программы. Выведите на экран окно Layout командой View, Form Layout Window. Запустите программу и закройте окно – в окне формы Layout появится значок формы, переместите его в нужное место экрана.
Mouse Pointer, Mouse Icon	0 (None)	Установка формы курсора мыши. Имеется 17 значений. Наиболее часто используют 11 и 13. 11 – песочные часы, 13 – стрелка с песочными часами. Если установить значение свойства Mouse Pointer 99, то можно использовать любой значок.
Visible	True	Видимость формы на экране. True – форма видима, False – невидима.

Продолжение таблицы 4.11

Windows-State	0	Определяет вид формы после загрузки. 0 – нормальный; 1 – форма уменьшается до значка; 2 – форма развернута на весь экран, соответствует операции Maximize.
ScaleMode	1 - Twip	Позволяет изменять единицу измерения масштаба. Существует семь вариантов: 0 – собственное значение, 1 – твипы, 3 – пиксели, 6 – мм, 7 – см.
ScaleHeight ScaleWidth	3195 4680	Используют, когда установлена не стандартная единица измерения масштаба. Установка данных значений приводит к присвоению свойству ScaleMode значения 0.
ScaleLeft ScaleTop	0 0	Описывают значения координат левой и верхней рамок формы относительно экрана.
ForeColor BackColor	Согласно настройкам Windows	Цвет текста и цвет фона, соответственно. Можно установить собственные значения, выбрав из списка. Закладка Palette выводит панель палитры. Закладка System выводит список текущих значений цвета различных элементов Windows.

События формы

Формы могут распознавать более 20 различных событий.

События ***Click***, ***DoubleClick*** служат для обработки одиночного и двойного щелчка мыши.

Как уже известно, с каждым событием, связана процедура – обработчик событий. События генерируются в ответ на действия пользователя – внешние события или генерируются системой - системные события. Например, реакция на щелчок мыши по форме реализуется в виде процедуры обработки события Click формы:

```
Private Sub Form_Click ()
    <текст программы>
End Sub.
```

Основные события формы, как правило, обрабатываются в таком порядке: Initialize, Load, Activate, Deactivate, Query Unload, Unload, Terminate.

Событие ***Initialize*** используется, обычно, для подготовки приложения к работе. В обработчике этого события переменным присваиваются начальные значения, расставляют элементы управления на форме, устанавливается масштаб для представления графических изображений. Данное событие возникает в момент создания экземпляра формы (до ее загрузки или отображения). Однако оно генерируется лишь один раз в течение всего сеанса работы приложения.

Поэтому в обработчик этого события нельзя помещать текст программы, который должен выполняться при каждой загрузке формы.

> *Если какой-то код должен выполняться несколько раз, то его нельзя помещать в обработчик данного события.*

Событие Initialize генерируется, например, при загрузке формы, её показе, а также при возвращении значения свойства (то есть какому-то свойству объекта закрытой формы присваивается программным путем некоторое значение) или вызове метода, определённого в форме. Например, событие Initialize генерируется при вводе команд:

```
    frmMyForm.Show
или
    Load frmMyForm
```

Переменные уровня формы доступны всем процедурам внутри программного модуля данной формы. После инициализации эти переменные существуют, пока выполняется приложение, даже если соответствующая форма выгружена.

Событие ***Load*** используется для выполнения каких-либо действий до вывода формы на экран. Оно также позволяет присвоить исходные значения свойствам формы и её элементам управления. Это событие возникает при каждой загрузке формы в память. При первой загрузке событие Load следует за событием Initialize. Событие Load генерируется также в результате применения метода Load или Show, а также после ссылки на свойства, методы или элементы управления незагруженной формы.

Пример 4.18. Установка размеров и позиционирование формы при загрузке

```
Private Sub Form_Load ( )
   Me.Height = 3100
   Me.Width = 4600
   Me.Top = (Screen.Height - Me.Height) / 2 ' центрирование формы
   Me.Left = (Screen.Width - Me.Width) / 2
End Sub
```

События ***Activate / Deactivate***. Эти события возникают при работе с несколькими формами. Событие *Activate* возникает, когда фокус ввода переходит на данную форму от другой формы того же приложения. При этом форма должна быть видима. Например, форма, загруженная оператором Load, остается невидимой, пока не примените метод Show или не установите свойство Visible формы как True. Событие Activate генерируется от события GotFocus - установка фокуса. Событие *Deactivate* происходит тогда, когда фокус ввода переходит с данной формы на другую форму этого же приложения.

Событие ***QueryUnLoad*** полезно, если нужно узнать, как именно пользователь закрывает форму. Данное событие происходит перед событием *Unload*. Событие QueryUnLoad происходит в следующих случаях:

- из системного меню формы выбрана команда Close;
- в программе выполняется оператор Unload;
- закрывается дочерняя MDI-форма, так как закрывается основная MDI-форма;
- завершается текущий сеанс работы в операционной системе Windows;
- Task Manager операционной системы Windows закрывает данное приложение.

Причина закрытия формы сохраняется в системной переменной UnloadMode. Если причина закрытия формы иная, то необходимо запросить согласие пользователя.

Пример 4.19. Использование события QueryUnload для контроля за закрытием формы.

```
Private Sub Form _QueryUnload(Cancel As Integer, UnloadMode As Integer)
   If  UnloadMode<>vbFormCode Then
         MsgBox "Если хотите выйти, то нажмите кнопку Close"
         Cancel = True        'форма остаётся открытой'.
   End If
End Sub.
```

Аргумент UnloadMode возвращает код причины закрытия формы. Системная переменная vbFormCode хранит допустимые значения кодов закрытия. Чтобы предотвратить выгрузку формы, аргументу *Cancel* присваивается значение *True* и форма остается открытой до принятия решения пользователем.

Событие **Unload** генерируется перед событием *Terminate*. Обработчик события Unload также можно использовать для проверки того, надо ли выгрузить форму или для определения операций, выполняемых при выгрузке формы. Можно также включить программу проверки уровня формы, необходимую для закрытия формы, или сохранения данных в файле. В обработчик события Unload можно добавить оператор End – он гарантирует выгрузку всех форм до завершения программы.

Присвоение аргументу Cancel любого ненулевого значения предотвращает удаление формы, но не запрещает другие события вроде выхода из среды Windows. Чтобы не допустить выхода из Windows, необходимо использовать событие QueryUnload.

Событие **Terminate** генерируется, когда из памяти удаляются все ссылки на экземпляр формы. Чтобы убрать из памяти переменные этой формы и освободить занимаемые системные ресурсы, присвойте объектной переменной формы значение Nothing:

Set frmMyForm = Nothing

Для всех объектов, кроме классов, событие Terminate генерируется после события Unload.

Методы формы

Метод выполняет над объектом какую-либо операцию. Знание методов форм позволяет разработать приложение, эффективно использующее системные ресурсы компьютера. Для управления формами в программах на Visual Basic предназначены методы: Load, Unload, Hide, Show.

Метод ***Load*** инициализирует и загружает форму в память, не выводя ее на экран. Это не относится к стартовой форме, стартовая форма всегда появляется на экране. *Любая ссылка на форму вызывает автоматическую загрузку ресурсов этой формы*, если они еще не загружены в память:

```
Load frmMyForm     'форма загружается, но не выводится на экран'
```

Метод ***Unload*** удаляет форму из памяти. Для ссылки на текущую форму можно использовать константу ***Me***.

```
Unload frmMyForm
```

Или

```
Unload Me
```

Метод ***Hide*** – убирает форму с экрана, не удаляя ее из памяти. Хотя элементы управления скрытой формы не доступны пользователю, к ним можно обращаться программно. Когда форма скрыта, пользователь не может взаимодействовать с соответствующей частью приложения.

Если на момент вызова метода Hide форма еще не загружена в память, она загружается, но на экране не появляется.

```
frmMyForm.Hide  или Me.Hide
```

Метод ***Show*** выводит форму на экран. Если на момент вызова оператора форма еще не загружена в память, то VB сначала вызывает метод Load. Метод Show позволяет показывать формы либо как модальные, либо как не модальные. Если на экран выводится модальная форма, то весь ввод с клавиатуры или с помощью мыши будет относиться только к модальной форме и работают лишь процедуры из модуля этой формы.

а) показать форму:

```
frmMyForm.Show
```

или

```
Me.Show
```

б) показать модальную форму

```
frmVvodData.Show vbModal
```

или

```
frmVvodData.Show 1
```

4.5.3. УПРАВЛЕНИЕ ФОРМАМИ

Добавление формы в проект

Добавление формы в проект осуществляется командой *Project, Add Form*. После ввода команд появится диалоговое окно ***Add Form***. Щелкнуть мышкой по значку *Form*, затем – по кнопке *Open*.

В проект будет добавлена новая форма.

Установка стартовой формы

По умолчанию первая форма в проекте считается стартовой. Чтобы сменить стартовую форму необходимо:

- выбрать из меню ***Project*** команду ***Project1 Properties***. Появится диалоговое окно *Project1 Properties*;
- раскройте список ***Startup Object***, выберете имя формы, которую хотите сделать стартовой и щелкните по кнопке ***Ок***.

Печать формы

Для простейшего вывода на печать всего содержимого формы имеется команда ***PrintForm.***

```
Private Sub Form_Click ()
        PrintForm
End Sub.
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие виды форм используются в Visual Basic, каково их назначение?
2. Назовите основные свойства формы.
3. Перечислите основные события формы.
4. Как установить размеры формы программным путем?
5. В какой последовательности генерируются события формы?
6. Для какой цели используются обработчики событий Load и Unload?
7. Перечислите основные методы формы и их назначение.
8. Каким образом можно установить причину закрытия формы?
9. Как удалить переменные закрываемой формы из оперативной памяти компьютера?
10. Как объявить форму модальной, каковы ее свойства?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились более подробно с формой, ее свойствами и методами. Знание их позволит более эффективно управлять формой и использованием ресурсов компьютера. При открытии и закрытии формы ее события наступают в определенной последовательности: Initialize, Load, Activate, Deactivate, Query Unload, Unload, Terminate. События Initialize, Load используются для установки начальных значений параметров элементов управления и значений переменных. События Query Unload и Unload используются для контроля закрытия формы. Методы формы Load, Show позволяют управлять ее открытием, метод Unload - закрытием, а метод Hide позволяет скрыть форму, не выгружая ее из памяти.

4.5.4 MDI – ФОРМА

При разработке приложений для повышения их функциональности и удобства использования разрабатывается много форм. Организацию взаимодействия с этими формами удобно осуществлять с помощью системы меню, помещенных в одну из форм – MDI-форму.

MDI–форма - это многодокументный интерфейс, предназначенный для организации взаимодействия нескольких независимых форм.

MDI-форма является родительской формой или контейнером для других форм (дочерних). В MDI-форме можно размещать только элементы управления, имеющие свойство выравнивания, такие как окно с рисунком PictureBox, картинка – Image. Можно поместить непосредственно в MDI-форму фоновое изображение.

MDI-формы применяются чаще всего для обслуживания однородных форм. Примерами их использования являются редактор Word или электронная таблица Excel. Но они с успехом могут применяться для организации взаимодействия и разнотипных форм.

Создание MDI-формы

Создание MDI-формы осуществляется командой ***Project,Add MDI Form.***

При запуске программы с MDI-формой программа автоматически устанавливает размеры дочерних окон, которые могут оказаться меньше, чем при настройке и поэтому часть объектов активной формы может быть невидимой. Чтобы избавиться от этого недостатка, необходимо в обработчике события Load каждой формы явно указать размеры и положение формы в окне, например:

```
Private Sub Form_Load()
    Me.Height = 2745
    Me.Width = 3090
    Me.Top = (MDIForm1.ScaleHeigth – Me.Heigth) / 2
    Me.Left = (MDIForm1.ScaleWidth – Me.Width) / 2
End Sub
```

Если одна из форм максимизируется, то и все последующие открываемые формы будут развернуты на все окно. Чтобы этого не происходило, необходимо при закрытии максимизированной формы приводить ее размеры в нормальное состояние. Для этого в обработчик события кнопки, предназначенной для закрытия формы, надо поместить следующий код:

```
Private Sub cmdExit_Click()
    Me.WindowState = 0
    Unload Me
End Sub
```

Работа с дочерними формами

Чтобы обычная форма стала подчиненной MDI-форме (дочерней формой), значение ее свойства ***MDIChild*** необходимо установить в True.

Дочерние формы показываются с помощью метода ***Show***, например:

```
Private Sub mnuVvod_Click()
    frmVvod.Show
End Sub
```

Дочерние формы могут иметь собственное меню. При развертывании дочерней формы ее заголовок заменит заголовок родительской формы, а меню дочерней формы заменит меню родительской формы.

Каждое приложение, имеющее MDI- форму, должно иметь пункт меню ***Окно***, позволяющее пользователю выводить дочерние окна каскадом (*Cascade*) или в виде мозаики (*Tile*). При размещении каскадом не минимизированные формы размещаются так, чтобы каждая предыдущая форма немного выступала из - за следующей. При размещении в виде мозаики формы могут размещаться горизонтально (Horizontal) или вертикально (Vertical).В первом случае каждая неминимизированная форма принимает ширину, рав-

ную ширине родительской формы, во втором случае каждая неминимизированная форма принимает высоту родительской формы. Для удобства управления открытыми формами в меню необходимо включать список открытых дочерних форм. Чтобы список открытых дочерних форм формировался автоматически, необходимо при разработке меню для элемента меню *Окно* установить флажок *WindowList*.

Управление размещением дочерних форм осуществляется с помощью метода ***Arrange***. Кроме того, VB имеет четыре константы для управления окнами: ***VbCascade*** – размещение окон каскадом, ***VbTileHorizontal*** – размещение окон горизонтально, ***VbTileVertical*** - размещение окон вертикально, ***VbArrangeIcons*** – пиктограммы всех минимизированных окон располагаются по нижнему краю родительской формы.

Для управления размещением открытых окон в меню Окно требуется ввести элементы меню второго уровня: *каскадом, горизонтально, вертикально,* - а затем поместить в обработчики события Click этих пунктов меню текст программы следующего вида:

```
имяMDI-формы.Arrange константаVisualBasic
```

Например:

```
Sub mnuCascade_Click
        MDIForm1.Arrange VbCascade
End Sub
Sub  mnuHorizontal_Click
        MDIForm1.Arrange VbTileHorizontal
End Sub
Sub  mnuVertical_Click
        MDIForm1.Arrange VbTileVertical
End Sub
```

Свертывание открытых окон в значок осуществляется щелчком мыши по кнопке закрытия окна.

4.5.5. РАЗРАБОТКА МЕНЮ ПОЛЬЗОВАТЕЛЯ

Многоуровневые меню

Разработка меню позволяет сделать приложение с более дружественным интерфейсом. Практически любая программа, написанная для Windows, содержит многоуровневые меню или меню, в которых команды сгруппированы по логическому назначению. При запуске программы в строке меню диалогового окна обычно видны только элементы верхнего уровня. При щелчке мышью по пункту меню открывается меню второго уровня и т.д.

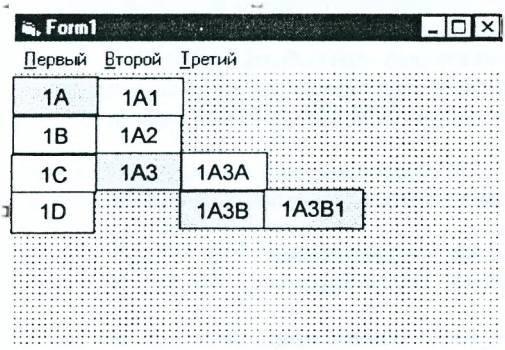


Рис. 4.27. Структура многоуровневого меню

VB позволяет иметь до шести уровней вложенности меню. Большое количество уровней тоже не совсем удобно. Рекомендуется использовать не более 3-х уровней вложенности. На рис. 4.27 представлено четырехуровневое меню. Главное меню: Первый, Второй, Третий. Меню первого уровня для меню Первый –1А, меню второго уровня – 1А1-1А3; меню третьего уровня – 1А3А – 1А3В, меню четвертого уровня – 1А3В1.

Средства для разработки меню

Меню любой конфигурации можно создать с помощью командных кнопок или с помощью текстовых полей. Для создания удобного меню с большой функциональностью потребуется приложить немало усилий. Однако эту задачу можно облегчить, если воспользоваться стандартными средствами VB.

Visual Basic 6.0 имеет удобное средство для разработки меню – редактор ***Menu Editor***, который вызывается командой ***Tools,Menu Editor*** или комбинацией клавиш ***CTRL – E.***

Диалоговое окно редактора Menu Editor приведено на рис. 4.28.

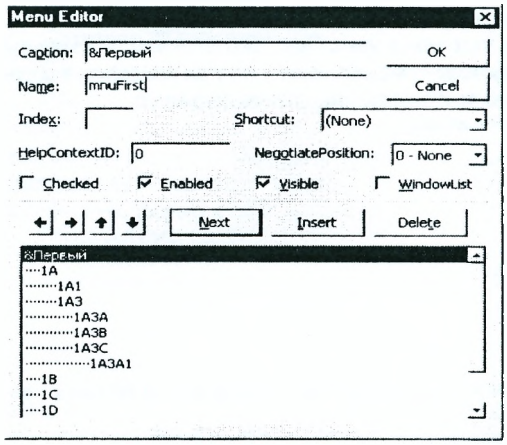


Рис. 4.28. Редактор меню Menu Editor

Строка ввода ***Caption*** служит для ввода наименования пункта меню, выводимого на экран. После нажатия клавиши OK или щелчка мыши введенное наименование появляется в окне редактора. Если перед одним из символов наименования пункта меню поставить символ "&" - амперсенд, то появится возможность вызывать пункт меню по нажатию данной клавиши (горячей клавиши) в комбинации с клавишей Ctrl. Символ, перед которым стоит знак "&", подчеркивается.

Строка ввода ***Name*** служит для ввода имени пункта меню, которое будет использоваться в программе для обработки событий. Перед именем пункта меню рекомендуется ставить префикс mnu, например, mnuFile. Программа не позволит пользователю выйти из редактора, пока всем пунктам меню не будут присвоены имена.

Окно ***Index*** используется в том случае, если имеется несколько пунктов меню с одинаковыми именами или надо сделать пункты меню частью массива элементов управления.

Окно ***Shortcut*** позволяет назначить каждому пункту меню комбинацию клавиш для быстрого вызова команд меню: Ctrl + клавиша, Shift + клавиша и др. При открытии списка появится список быстрых клавиш, из которого надо выбрать нужный.

Окно ***HelpContextID*** обеспечивает ввод идентификатора, который используется в электронной справочной системе для выдачи контекстно-зависимой справки по вашему приложению.

Окно ***NegotiatePosition*** – служит для определения способа отображения меню на экране, когда один из связанных объектов приложения активен: не показывать, слева, справа, по центру.

Флажок ***Checked***. Если значение данного свойства равно True, то возле соответствующего пункта меню появляется галочка. Это сигнализирует о том, что соответствующий параметр выбран.

Флажок ***Enabled***. Определяет доступность данного пункта меню. Если его значение равно False, то пункт меню недоступен.

Флажок ***Visible***. Данное свойство определяет, будет ли виден на экране соответствующий пункт меню. При разработке приложения можно предусмотреть несколько наборов меню, которые должны появляться на экране в соответствующие моменты времени. Например, если в приложении не открыто ни одно окно, меню Window не должно появляться на экране.

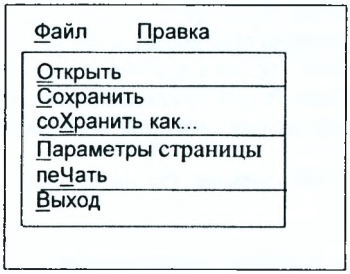


Рис. 4.29. Группировка пунктов меню

Флажок ***WindowList***. Данный флажок служит для автоматического формирования списка открытых окон, являющихся элементами многодокументного интерфейса (MDI). Установка этого флажка для элемента управления меню верхнего уровня приведет к тому, что в данном элементе будет автоматически формироваться динамический список всех активных дочерних окон.

Кнопка ***Next*** предназначена для добавления новых пунктов меню.

Кнопка ***Insert*** позволяет вставить поле для ввода нового пункта меню.

Кнопка ***Delete*** служит для удаления выделенного пункта меню.

Группировка элементов списка пункта меню

Пункты меню, близкие по назначению целесообразно группировать, отделяя их от других пунктов меню горизонтальной чертой – разделительная линия (Separator Bar) (рис.4.29). Эта черта создается так же, как и другие пункты меню, но вместо наименования пункта меню (свойство Caption) вводится дефис (-). Имя данному пункту меню можно присвоить произвольно, например, mnuRaz1 и т. д.

Управление размещением пунктов меню

В редакторе *Menu Editor* изменение уровня вложенности элемента меню осуществляется с помощью кнопок ➡ и ⬅. Первая кнопка понижает уровень, вторая – повышает. Кнопки ⬆ и ⬇ служат для перемещения выделенного пункта меню по вертикали. Уровень вложенности элемента управления при этом не изменяется.

Взаимодействие меню MDI-формы и дочерних форм

Дочерние формы, так же как и MDI-форма (родительская), могут иметь меню, созданные с помощью редактора *Menu Editor.* При открытии дочерней формы, содержащей такое меню, оно замещает меню родительской формы. При этом часть пунктов меню MDI-формы может выступать из-под меню дочерней формы, что может вызвать недоразумения. Поэтому программист должен позаботиться о том, чтобы сделать пункты меню родительской формы невидимыми, пока открыта дочерняя форма, а после ее закрытия сделать их снова видимыми.

Использование пунктов меню

Для элементов управления меню предусмотрено только одно событие – Click. Оно возникает, когда пользователь с помощью мыши или клавиатуры выбирает нужный пункт меню. Например, для открытия формы с помощью пункта меню можно записать следующий текст программы:

```
Private Sub mnuOpen_Click()
        frmForm1.Show
End Sub
```

В процедуре обработчика события пункта меню может быть записан, естественно, текст любой программы, которая должна быть выполнена при выборе данного пункта меню.

4.5.6. КОНТЕКСТНОЕ МЕНЮ

Контекстное (всплывающие) меню появляется, обычно, после щелчка правой кнопкой мыши по объекту. Порядок разработки контекстного меню практически ничем не отличается от порядка разработки обычного меню. Отличие состоит в том, что для меню верхнего уровня свойство Visible устанавливается в False. То есть, в исходном состоянии меню верхнего уровня, а следовательно, и подчиненные ему элементы меню нижних уровней, невидимы.

Для вызова контекстного меню используется метод ***PopupMenu***. Синтаксис команды вызова всплывающего меню:

```
ИмяФормы.PopupMenu ИмяЭлементаМеню
```

Команда вызова контекстного меню записывается в обработчик события ***MouseUp*** соответствующего объекта (отпускание клавиши мыши после щелчка мышью по объекту).

Пример 4.20. Требуется разработать контекстное меню для изменения высоты шрифта.

Рассмотрим для примера меню настройки высоты шрифта для формы. Откроем новую форму и создадим меню следующей структуры (элементы меню отделены друг от друга точкой с запятой): *Высота шрифта; ...8; ...12; ...14; ...18; ...24*. Для элемента меню "Высота шрифта" введем имя mnuShriftHeight и снимем флажок Visible. Пунктам меню второго уровня присвоим имена mnu8, mnu12 и т. д.

Напишем текст программы.

В обработчик события Click формы запишем оператор печати текста:

```
Private Sub Form_Click()
    Print "Привет"
End Sub
```

В обработчик события MouseUp формы запишем программу проверки условия нажатия правой клавиши мыши:

```
Private Sub Form_MouseUp(Button As Integer, _
    Shift As Integer, X As Single, Y As Single)
    If Button = vbRightButton Then
        Form1.PopupMenu mnuShriftHeight
    End If
End Sub
```

Системная переменная Button возвращает код нажатой клавиши мыши и ее значение сравнивается с константой vbRightButton – код правой клавиши мыши, переменная Shift – возвращает код нажатых клавиш Ctrl, Alt или Shift, переменные X и Y определяют координаты для вывода контекстного меню на экран – координаты указателя мыши.

В обработчики событий Click пунктов меню второго уровня запишем следующие тексты программ:

```
Private Sub mnu8_Click()
    Form1.FontSize =8
End Sub
Private Sub mnu14_Click()
    Form1.FontSize = 14
End Sub
Private Sub mnu24_Click()
    Form1.FontSize = 24
End Sub
```

```
Private Sub mnu12_Click()
    Form1.FontSize = 12
End Sub
Private Sub mnu18_Click()
    Form1.FontSize = 18
End Sub
```

Запустим программу. При щелчке левой клавишей мыши по форме появляется текст "Привет" с текущим значением высоты шрифта. Щелкнем правой клавишей мыши по форме – появится контекстное меню формы со списком высоты шрифта. Выберем нужную высоту шрифта щелчком мыши – появится текст с выбранной высотой шрифта.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое MDI-форма, чем она отличается от обычной формы?
2. Как сделать форму дочерней?
3. Как установить стартовую форму?
4. Как разрабатывается меню пользователя?
5. Расскажите назначение окон ввода и флажков окна диалога Menu Editor.
6. Как используется меню?
7. Как обеспечить управление открытыми окнами программы?
8. Что необходимо сделать, чтобы список открытых форм формировался автоматически?
9. Чем отличается порядок разработки контекстного меню от порядка разработки обычного меню?
10. Как используется контекстное меню?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с MDI-формой и ее основными свойствами: MDI – форма не является контейнером, то есть в нее нельзя помещать другие объекты. Она служит для управления простыми формами.

Удобным средством управления проектом является меню. Правильно разработанное меню определяет функциональность разрабатываемого приложения. Через меню пользователь должен иметь возможность обратиться к любой функции приложения. Меню, как правило, имеет иерархическую структуру. Глубина вложения пунктов меню не должна превышать шести уровней.

Для выполнения функций, связанных с некоторыми элементами управления формы или самой формы, целесообразно разрабатывать контекстные меню. Порядок разработки контекстного меню не отличается от порядка разработки главного меню. Единственное отличие состоит в том, что свойству Visible пункта меню первого уровня контекстного меню присваивается значение False.

Способ использования контекстного меню существенно отличается от способа использования главного меню.

4.6. СТАНДАРТНЫЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ VB

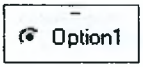
При разработке интерфейса программы пользователя до настоящего момента нами использовались только Надписи, Текстовые поля, Командные кнопки и рамки. Однако в большинстве случаев при разработке пользовательского интерфейса этих элементов управления может оказаться недостаточно. На панели инструментов *Toolbox* имеется ряд элементов управления, которые позволяют улучшить интерфейс. Это такие элементы управления, как *флажки, переключатели, списки, таймер, линейки прокрутки, список устройств, список каталогов,строка состояния, список файлов и др.*

4.6.1. ФЛАЖКИ И ПЕРЕКЛЮЧАТЕЛИ

Флажки (CheckBox) – это элементы управления, которые можно отмечать "галочками". Флажки, независимо от их числа и места установки, независимы друг от друга. Флажки можно применять как самостоятельно, та и группами.

Имя объекта по умолчанию – Check1, префикс – chk.

Важнейшим событием элемента CheckBox является событие ***Click***, а основным свойством – его значение - ***Value***. С помощью этого свойства можно определить состояние объекта. Свойство Value может иметь три значения: 0 – не отмечен; 1- отмечен; 2 – отмечен, но недоступен. Последнее значение может быть установлено только программно. Наименование объекта (свойство Caption) заносится непосредственно в объект, поэтому при установке объекта на форму необходимо предусмотреть достаточно места для ввода наименования объекта.



Переключатели (OptionButton) – это элементы управления, которые можно отмечать точкой, выбирая один элемент из группы. Переключатели могут применяться как самостоятельно, так и группами. При установке на форму нескольких переключателей они автоматически объединяются в группу. При этом в группе может быть включенным только один элемент. На форме можно организовать несколько групп переключателей, поместив их в рамку (Frame).

Имя объекта по умолчанию Option1, префикс - opt

Переключатели имеют те же свойства, что и флажки. Для удобства управления эти элементы управления целесообразно объявлять как массив управляющих элементов.

Для контроля состояния флажков и переключателей и управления программой можно использовать функции If/ElseIf/Else/End If и Select Case.

Рассмотрим примеры применения этих элементов.

Пример 4.21. Использование элемента CheckBox (рис. 4.30).

Поместим на форму кнопку, четыре флажка и создадим из них массив элементов управления. Напишем текст программы в обработчики событий Load формы и обработчик события Click кнопки Печать:

```
Option Explicit
Dim S(3) As String
-----------------------------------
Private Sub cmdPrint_Click()
Dim i As Integer
  Print
  For i = 0 To 3
    If Check1(i).Value Then
      Print Tab(5); S(i)
    End If
  Next i
End Sub
-----------------------------------
Private Sub Form_Load()
  S(0) = "Литература"
  S(1) = "Искусство"
  S(2) = "Спорт"
  S(3) = "Музыка"
End Sub
```

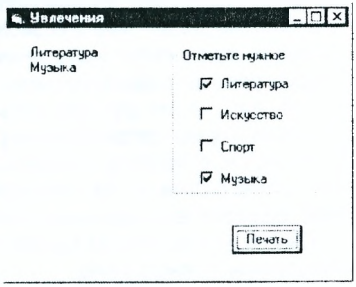


Рис. 4.30. Использование флажков

Пример 4.22. Использование индекса элемента OptionButton (рис.4.31)

Установите на форму три переключателя и создайте из них массив элементов управления. В обработчике со-

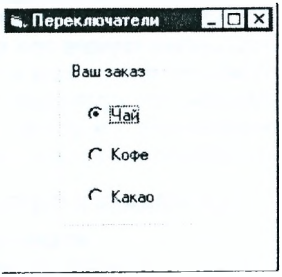


Рис.4.31. Переключатели

бытия Click элемента управления Option1 параметром является системная переменная Index, которая возвращает номер активного элемента управления. Значение параметра обрабатывается с помощью оператора Select Case. Для управления используется один обработчик события.

```
Private Sub Option1_Click(Index As Integer)
  Select Case Index
    Case 0
      MsgBox "Вы выбрали чай"
    Case 1
      MsgBox "Вы выбрали кофе"
    Case 2
      MsgBox "Вы выбрали какао"
  End Select
End Sub
```

4.6.2. СПИСКИ И ПОЛЯ СО СПИСКАМИ

Имеется несколько типов полей со списками: простые списки (*ListBox*), раскрывающиеся списки или поле со списком (*ComboBox*), а также элемент *ImageCombo*.

Списки позволяют выбирать значения из списка, вносить записи в процессе работы программы и на этапе разработки. Если данные не умещаются в окне, то автоматически появляется линейка прокрутки.

Основные свойства и методы списков

При установке объекта ListBox на форму ему по умолчанию присваивается имя List1. Префикс для имени объекта – lst.

Свойства объекта List

Основными свойствами списка являются:

Text, List, ListIndex, ListCount, Columns, Sorted, ItemData, MultiSelect.

Text – хранит значение выбранного элемента списка, так же как и у объекта TextBox;

List – это свойство хранит все значения списка. Все записи в списке имеют индекс (как массивы), нумерация элементов списка начинается с нуля. Зная индекс элемента, можно выбрать его из списка. Синтаксис команды:

```
<переменная>=lstList1.List(i)
```

Данные в список можно вносить как на этапе разработки, так и в процессе работы программы.

Для добавления элемента в список на этапе разработки введите его в строке свойства List и нажмите Ctrl+Enter для перехода на новую строку.

ListIndex - возвращает индекс элемента списка:

```
< индекс_элемента >= lstList1.ListIndex
```

Если в списке не выбран ни один элемент, то значение свойства ListIndex равно –1. Можно комбинировать свойства List и ListIndex. Например:

```
Index=lstList1.ListIndex
<Элемент_списка>=lstList1.List(Index)
```

или

```
<Элемент_списка>=lstList1.List(lstList1.ListIndex)
```

Тот же результат получим при использовании свойства Text:

```
<Элемент_списка>=lstList1.Text
```

ListCount – сохраняет текущее значение числа элементов списка.

Columns – это свойство позволяет в процессе разработки отображать данные в не-

сколько столбцов. Заполнение столбцов в этом случае осуществляется последовательно – сначала заполняется первый столбец, затем второй и т.д.

Sorted – определяет способ расположения элементов в списке. Значение свойства Sorted устанавливается только на этапе разработки программы. Если это свойство установлено в *True*, то все элементы списка будут сортироваться по алфавиту, даже если они добавлены с указанием индекса. Индекс последнего добавленного элемента имеет свойство ***NewIndex***.

Вновь добавленный элемент имеет и другое интересное свойство списка – ***ItemDate***(). С помощью этого свойства каждому элементу списка можно поставить в соответствие число типа *Long* (целое двойной длины). Используя это свойство, можно составить список сотрудников, сохранив их индивидуальные номера в свойстве ItemData:

```
lstPersonal. AddItem "Иванов П. С."
lstPersonal.ItemData(lstPersonal.NewIndex)=8763
```

объект свойство индекс

Правда, индивидуальный нормер можно присвоить и другим способом:

lstList1.ItemData (List1.ListIndex)=10986

MultiSelect – это свойство позволяет выбирать одновременно несколько элементов. Оно имеет три значения: 0 – множественный выбор невозможен; 1 – простой множественный выбор; 2 – расширенный выбор.

При простом множественном выборе можно выделять произвольные строки при нажатой клавише Shift или Ctr. Для отмены выделения необходимо повторно щелкнуть мышью по каждому выделенному элементу списка.

При рассширенном множественном выборе элементов списка при выделении непрерывной группы элементов списка необходимо установить курсор на первый выделяемый элемент списка, нажать и удерживать клавишу Shift, щелкнуть мышью последний выделяемый элемент списка; при выделении группы разрозненных элементов списка выделить первый элемент списка, нажать и удерживать клавишу Ctrl, щелкнуть мышью по другим выделяемым элементам списка.

Для отмены выделения щелкните мышью по любому полю.

При множественном выборе свойство Text содержит текст последнего выбранного элемента списка или нет.

Selected – это свойство позволяет определить выделен данный элемент списка или нет:

<Логическая_переменная>=lstList1.Selected

Если элемент списка выделен, то возвращается значение True.

Методы объекта List

AddItem - добавление элементов в список:

lstList1.AddItem <Элемент [, Индекс]>

lstList1.AddItem "Персональная выставка",5

добавляется пятый элемент списка. Если номер элемента в списке не указывается, то он помещается в конец списка.

RemoveItem - удаление элементов из списка:

lstList1. RemoveItem ИндексЭлемента

Удалить пятый элемент списка:

lstList1.RemoveItem 5

Удалить выделенный элемент списка:

lstList1.RemoveItem lstList1.ListIndex

ListBox Clear - удаление всех элементов списка.

Поле со списком

Имя объекта по умолчанию Combo1, префикс – cbo.

Поле со списком обладает свойствами как списка, так и поля ввода.

Для выбора элемента списка используется событие ***Click***, а для изменения записи в поле ввода текста событие – ***Change***.

Для поля со списком важное значение имеет свойство **Style**, определяющее внешний вид и функционирование поля со списком. Оно может принимать три значения:

0 (значение по умолчанию) - текстовое поле и раскрывающийся список. В текстовое поле можно вносить данные;

1 – текстовое поле и постоянно открытый список;

2 – отличается от значения 0 тем, что пользователь не может вводить текст в текстовое поле.

Пример списков приведен на рис 4.32.

Установка начального значения

При загрузке списка в поле ввода обычно появляется какое-то значение. Это значение можно установить при разработке программы, присвоив свойству *Text* нужное значение, или установить его программно, используя свойство ListIndex, например:

```
Combo1.ListIndex = 2
```

Если вы хотите, чтобы при загрузке формы поле ввода осталось пустым, присвойте свойству *Text* на этапе разработки пустую строку.

Пример 4.23. Разработайте форму для исследования свойств элемента List (рис.4.32). Установите на форму элементы управления согласно рисунку, внесите данные в свойство List объектов и установите их свойства. Имена кнопок: Добавление – cmdAdd; Добавление с указанием номера cmdAppend; Удаление –cmdDel. Запишите текст программы:

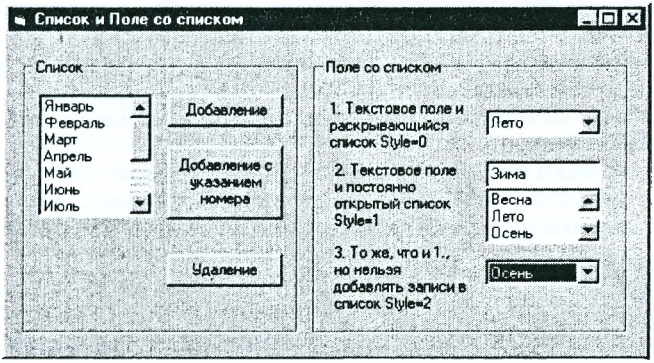


Рис. 4.32. Списки

```
Option Explicit
Dim S As String
---------------------------------------------------------
Private Sub cmdAdd_Click()
  S = InputBox("Введите название месяца", _
  "Добавление элемента в список")
  lstList1.AddItem S  ' добавление элемента в список
End Sub
```

В случае, если элемент списка вводится с указанием номера, текст программы усложняется:

```
Private Sub cmdAppend_Click()
    Dim S1 As String, n As Integer
    ' Ввод элементов списка
    S = InputBox("Введите элемент списка и его индекс через запятую", _
        "Добавление элемента в список")
    ' поиск, выделение номера элемента списка и индекса
    n = InStr(S, ",") ' поиск в строке текста запятой
    S1 = Left(S, n - 1)  ' выделение из строки А элемента списка
    n = Val(RTrim(Right(S, Len(S) - n))) ' выделение индекса и
                                         ' преобразование его в число
    lstList1.AddItem S1, n  ' добавление элемента в список
End Sub
-------------------------------------------------------------------------------
Private Sub cmdDel_Click() 'процедура удаления элемента списка
    lstList1.RemoveItem lstList1.ListIndex
End Sub
-------------------------------------------------------------------------------
Private Sub Form_Load()
   cboCombo1.ListIndex = 1
'свойству Text элемента cboCombo1 присваивается значение
'первого элемента списка
End Sub
```

4.6.3. ПОЛОСА ПРОКРУТКИ

Полосы прокрутки используются во всех списках, элементах управлениия TextBox, сетке, но могут выполнять и некоторые специфические свойства, например, роль регуляторов. По умолчанию имени объекта присваивается значение HScroll1 – для горизонтальной линейки прокрутки и VScroll1 – для вертикальной линейки прокрутки. Префикс – hsc.

С точки зрения программирования, полосы прокрутки являются одними из самых простых элементов управления.

Основными свойствами полосы прокрутки *ScrollBar* являются свойства *Max*, *Min*, *Value*, *SmallChange*.

Свойства ***Max*** и ***Min*** определяют диапазон измеряемых величин, который может изменяться от – 32768 до + 32767. Значение свойства ***Value*** напрямую зависит от установленного диапазона и определяется текущим положением ползунка.

Рис. 4.33. Полоса прокрутки

Свойство ***SmallChange*** определяет, на какую величину будет изменяться значение свойства Value при щелчке мышью по правой или левой кнопке полосы прокрутки.

Для полосы прокрутки важное значение имеют события ***Change*** и ***Scroll.*** При изменении положения ползунка автоматически возникает событие *Change* для описываемого элемента управления. Событие *Scroll* возникает перед событием Change.

Пример 4.24. Использование свойств полосы прокрутки (рис.4.33.).

Поместим на форму три полосы про-

крутки, элемент TextBox и шесть Надписей. Текстовое поле будем использовать для отображения цвета. Три Надписи задействуем для обозначения полос прокрутки, а остальные Надписи используем для отображения числового значения свойства Value полос прокрутки. Свойству Max элемента ScrollBar присвоим значение 255, свойству Min – ноль, а свойству SmallChange - 1.

Текст программы для полос прокрутки поместим в обработчик события Scroll, чтобы одновременно с перемещением ползунка менялось и числовое значение. При перетаскивании ползунка полосы прокрутки цвет текстового поля на экране будет синхронно изменяться, а в Надписях будет показываться значение свойства Value.

```
Private Sub VScroll1_Scroll()
    Label4(0).Caption = VScroll1.Value
    Text1.BackColor = RGB(VScroll1.Value, VScroll2.Value, VScroll3.Value)
End Sub
Private Sub VScroll2_Scroll()
    Label4(1).Caption = VScroll2.Value
    Text1.BackColor = RGB(VScroll1.Value, VScroll2.Value, VScroll3.Value)
End Sub
Private Sub VScroll3_Scroll()
    Label4(2).Caption = VScroll3.Value
    Text1.BackColor = RGB(VScroll1.Value, VScroll2.Value, VScroll3.Value)
End Sub
```

4.6.4. ЭЛЕМЕНТ УПРАВЛЕНИЯ SLIDER

Другим элементом управления, похожим на полосу прокрутки и встречающимся в приложениях Windows, является элемент ***Slider*** (рис. 4.34). Он позволяет выбирать дискретное значение или набор значений из определенного диапазона.

Рис.4.34. Элемент управления Slider

Этого элемента управления нет среди стандартных элементов управления панели ToolBox. Для загрузки его на панель элементов управления выберите команду ***Project, Components***, установите флажок ***Microsoft Windows Common Controls 6.0 (SP3)*** или ***Common Controls 5.0 (SP2)***.

Элемент Slider имеет свойства ***Min***, ***Max*** и ***Value*** как и полоса прокрутки. Параметры изменения значений при перемещении ползунка в области значений определяют свойства ***SmallChange*** – минимальное изменение и ***LargeChange*** – максимальное изменение.

В отличие от полосы прокрутки для этого элемента можно определить не только одно значение, но и некоторый диапазон значений. Для этого служат свойства ***SelStart*** и ***SelLength***, но само выделение диапазона должно выполняться программно. Новое свойство ***Text*** позволяет задавать текст надписи, которая будет отображаться при перемещении ползунка. Позиция отображения этой надписи определяется значением свойства ***TextPosition***. Новым является также и событие ***Validate***. В обработчике этого события помещается код проверки правильности введенных данных.

4.6.5. СЧЕТЧИК

Счетчик (элемент управления UpDown) используется для установки различных числовых значений. Данного элемента нет среди стандартных элементов управления. Для добавления его на панель элементов управления введите команду ***Project, Components***, установите флажок ***Microsoft Windows Common Con-***

trols-2 6.0. Элемент UpDown можно использовать без написания кода, если правильно определить свойства *AutoBuddy, SyncBuddy, BuddyControl и BuddyProperty* элемента управления Buddy.

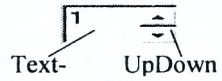


Рис.4.35. Композиция TextBox +UpDown

Buddy – это элемент управления, значение которого меняет UpDown (рис.4.35.). Buddy можно установить с помощью свойства *Buddy-Control* или свойства *AutoBuddy* (рис. 4.36) . В первом случае надо вручную записать в строку ввода имя элемента управления, значение которого будет менять элемент UpDown, во втором случае программа автоматически внесет в строку ввода имя элемента управления, у которого значение свойства TabIndex на единицу меньше, чем у UpDown.

Свойство *BuddyProperty* указывает, какое свойство элемента Buddy должно синхронизироваться со свойством *Value* элемента UpDown. Для элемента управления TextBox таким свойством будет свойство Text. Если свойству *BuddyProperty* присвоить значение Default, то автоматически будет обновляться стандартное свойство элемента управления, связанного с UpDown.

Свойство *SyncBuddy* – предписывает элементу управления UpDown обновлять значения связанного с ним элемента управления.

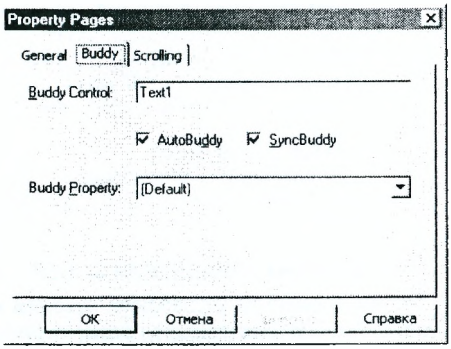


Рис.4.36. Настройка элемента UpDown

Закладка *General* позволяет устанавливать положение элемента управления слева или справа от связанного с ним объекта - свойство *Aligument* и ориентацию объекта: вертикально или горизонтально – свойство *Orientation.*

На закладке *Scrolling* устанавливается область значений с помощью свойств Min, Max, Value, Wrap.

Свойство *Wrap* – определяет, примет ли элемент управления свое минимальное значение, если щелкать мышью по кнопке со стрелкой вверх до тех пор, пока будет превышено максимально допустимое значение (и наоборот). Свойство *Increment* позволяет установить шаг изменения значения свойства Value при щелчке по кнопкам элемента управления UpDown.

Порядок работы по настройке элемента UpDown.

1. Установите на форму элемент TextBox и присвойте ему имя Text1.
2. Установите на форму элемент UpDown и присвойте ему имя UpDown1. Свойства *TabIndex* этих элементов будут смежными.
3. Щелкните правой кнопкой мыши по элементу UpDown и выбрите в контекстном меню опцию Properties – откроется диалоговое окно (Property Pages) настройки свойств элемента UpDown.
4. Откройте закладку Buddy (рис. 4.36) диалогового окна и установите флажок Auto-Buddy. В строке ввода автоматически устанавливается имя элемента управления Text1.
5. Установите свойство BuddyProperty равным Default. При этом автоматически устанавливается флажок SyncBuddy - обновлять стандартное свойство Text элемента Text1.

6. Откройте закладку Scrolling и установите значения Min, Max и Increment.

7. Настройте, при необходимости, ориентацию элемента UpDown с помощью закладки General диалогового окна.

8. Завершите работу, щелкнув по кнопке OK.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните назначение элементов управления CheckBox и OptionButton. В чем состоит отличие в использовании названных элементов управления?
2. Назовите основные свойства элементов управления CheckBox и OptionButton.
3. Приведите фрагмент программы для использования свойства Value элементов управления CheckBox и OptionButton.
4. Как создать массивы элементов управления CheckBox и OptionButton. Приведите фрагмент программы для использования свойства Index этих элементов управления.
5. Для каких целей применяются списки List и ComboBox? В чем отличие списка и поля со списком?
6. Перечислите основные свойства поля со списком ComboBox. Приведите синтаксис использования свойств элемента ComboBox.
7. Перечислите основные методы поля со списком ComboBox. Приведите синтаксис использования методов элемента ComboBox.
8. Как занести данные в список на этапе разработки формы?
9. Как дополнить список элементов управления ComboBox или List в процессе работы?
10. Для чего предназначена полоса прокрутки ScrollBar?
11. Назовите основные свойства полосы прокрутки.
12. В чем отличие элемента управления ScrollBar от элемента управления Slider?
13. Каково назначение элемента управления UpDown?
14. Как настроить элемент UpDown без написания программы?

ЗАКЛЮЧЕНИЕ

Элементы управления, рассмотренные в настоящем разделе позволяют разрабатывать разнообразный и дружественный интерфейс для форм и окон диалога путем:

- выбора вариантов решения задач с использованием флажков и переключателей;
- хранения и выборки данных из списков, предложения вариантов решения по умолчанию;
- наглядной установкой значений параметров вводом значений в текстовые поля с клавиатуры или с использованием мыши или перемещений ползунков регуляторов.

4.7. ДОПОЛНИТЕЛЬНЫЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

Одним из важных аспектов проектирования приложений пользователя является информирование пользователя о состоянии приложения и ходе выполнения программы. Например, в текстовых редакторах необходимо указывать номер текущей страницы и общее число страниц в документе, при работе с файлами данных пользователя могут интересовать сведения о размере файла, числе записей, номере текущей записи и т. д. В процессе работы пользователю может быть интересно знать, как идет загрузка программы, копирование файла или как долго будет идти расчет параметров. Для предоставления информации об интересующих пользователя данных и ходе протекания процессов могут использоваться такие средства Visual Basic, как

- строка состояния – StatusBar;
- панель инструментов – ToolBar;

- графическое представление хода выполнения программы – ProgressBar;
- анимационные видеоролики – Animation.

Одним из важных элементов интерфейса, позволяющих управлять программой, является панель инструментов, на которую выводятся пиктограммы наиболее часто используемых команд.

Элементов управления для реализации этих функций нет среди стандартных элементов управления. Поэтому их загружают с панели элементов Компоненты. Введите команду ***Project, Components*** и установить флажок возле компонента ***Microsoft Windows Common Controls 6.0 (SP3).*** Этот компонент содержит такие элементы управления, как Строка состояния – *StatusBar*, панель инструментов – *ToolBar*, индикатор процесса – *ProgressBar*, регулятор ползункового типа – *Slider*, список изображений - *ImageList*, дерево просмотра – *TreeView*, усовершенствованный список - *ListView*, усовершенствованное поле со списком – *ImageCombo*, стандартную панель инструментов приложения - *TabStrib*. Ограниченный объем учебного пособия не позволяет рассмотреть подробно все эти элементы, поэтому ограничимся кратким описанием лишь некоторых из упомянутых элементов управления.

4.7.1. СТРОКА СОСТОЯНИЯ

Создание строки состояния

Строка состояния - стандартный элемент управления Windows. После загрузки строки состояния на панель элементов управления установите ее на форму. Панель состояния может содержать до 16 отдельных панелей, в которых отображается различная информация, например, дата, время, имя файла и др.

Настройку строки состояния можно проводить с помощью окна Свойства среды разработки, а также с помощью панели свойств (Property Pages) (рис. 4.37). Для вызова панели свойств щелкните правой кнопкой мыши по строке состояния и выберите в меню пункт Properties. На экране появится панель свойств.

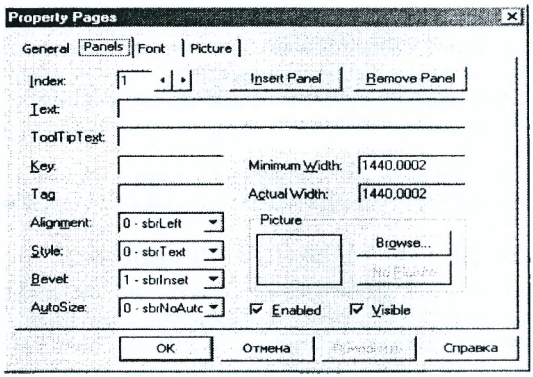


Рис. 4.37. Строка состояния

Закладка General позволяет установить общие свойства строки состояния. Прежде всего необходимо определить вид и расположение панелей. Эти параметры определяются с помощью свойств *Align* и *Style*.

Свойство ***Style*** определяет вид строки состояния. Строка состояния может быть простой текстовой (sbrSimple) – состоит из одной панели или составной (sbrNormal) – состоит

из нескольких самостоятельных панелей со своими свойствами. Панели строки состояния имеют индекс, как элементы массивов элементов управления, поэтому обращение к ним осуществляется с указанием индекса. Нумерация панелей начинается с единицы.

Свойство ***Align*** определяет, к какой стороне формы должна "прикрепиться" строка состояния. Это свойство может принимать пять значений: vbAlignTop – к верхнему краю формы; vbAlignBottom – к нижнему краю формы; vbAlignLeft – к левому краю формы; vbAlignRight – к правому краю формы; vbAlignNone – строка состояния может располагаться в любой части формы.

Закладка Panels позволяет настроить свойства каждой из 16 панелей строки состояния.

Основные свойства панелей строки состояния

Свойство ***Index*** определяет номер панели строки состояния.

Имеется восемь основных свойств панелей строки состояния, которые определяют их внешний вид и функции.

Text – определяет текст, который будет появляться в текстовой панели. Этот текст выводится обычно в процессе выполнения программы;

ToolTipText – текст подсказки, который появляется при зависании указателя мыши над панелью;

Alignment – Управляет выравниванием текста в панели: слева, справа или по центру;

Style – определяет тип создаваемой панели. Можно установить семь типов панелей: *sbrText*- позволяет отобразить текст или растровое изображение, указанные соответственно в свойствах Text или Picture панели строки состояния; *sbrCaps* – отображает состояние клавиши CapsLock; *sbrNum* - отображает состояние клавиши NumLock; *sbrIns* - отображает состояние клавиши Insert; *sbrScrl* - отображает состояние клавиши ScrolLock; *sbrTime* – отображает текущее время; *sbrDate* – отображает текущую дату.

Bevel – Определяет тип затенения для имитации объемности панели;

AutoSize – определяет принцип управления размером панели из программы;

MinWidth – устанавливает минимальные размеры панели;

Picture – позволяет поместить изображение в строку состояния.

Управление панелями строки состояния

На этапе разработки добавление и удаление панелей осуществляется с помощью кнопок: Insert Panel – добавление панели и Remove Panel – удаление панелей.

Для управления панелями строки состояния в процессе работы программы используются методы *Add*, *Remove* и *Clear.*

Add – добавить панель. Синтаксис метода:

```
Add ([Index] [,клавиша] [,текст] [, стиль] [,картинка]) As Panel
```

Remove – удаление панели. Синтаксис метода:

```
Remove [Index]
```

Clear – удаляет все панели из строки состояния. Синтаксис метода: Clear

Пример вывода текста в строку состояния динамически:

```
StatusBar1.Panels(1).Text ="Ждите. Идет решение задачи"
```

Пример 4.25. Управление строкой состояния

Требуется создать строку состояния с одной текстовой панелью; вывести в нее текст "Моя панель"; добавить в строку состояния одну текстовую панель, в которую поместить текст "Для продолжения нажмите любую клавишу" и панель для отображения текущего времени (рис. 4.38). Удалять первую текстовую панель по нажатию любой клавиши.

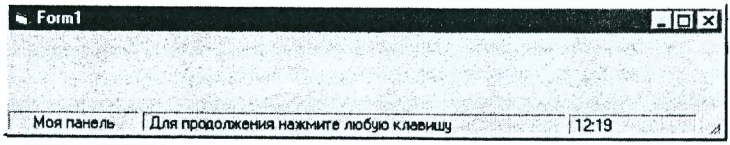


Рис. 4.38. Пример строки состояния

Порядок работы

- Введите команду Project, Components, установите флажок Microsoft Windows Common Controls 6.0 (SP3) и нажмите Ok.
- Установите на форму элемент управления StatusBar.
- Установите свойство Align строки состояния в панели Свойств равным 2 – прицепляется к нижнему краю формы, а свойство Style оставьте без изменения (sbrNormal).
- Вызовите панель Property Pages щелчком правой кнопки мыши по панели и откройте закладку Panels.
- Введите в поле Text текущей панели текст "Моя панель".
- Напишите в обработчики событий Load и KeyPress формы текст программы:

```
Private Sub Form_Load()
    Me.Width = 8000
    Me.Height = 1500
    Dim S As String
    With StatusBar1
        'добавляем текстовую панель в левую часть строки состояния
        .Panels.Add 2, , "Для продолжения нажмите любую клавишу", sbrText
        'настраиваем свойство AutoSize – автоматическая установка длины
        'строки по длине текста.
        .Panels(2).AutoSize = sbrSpring
        ' добавляем панель с текущим временем
        .Panels.Add , , , sbrTime
    End With
End Sub
Private Sub Form_KeyPress(KeyAscii As Integer)
    ' удаляем панель созданную по умолчанию
    With StatusBar1
        .Panels.Remove 1
    End With
End Sub
```

- Запустите программу и проверьте ее работу.

4.7.2. ИНДИКАТОР ПРОЦЕССА

Индикатор процесса (ProgressBar) предназначен для отображения медленно протекающих процессов. Этот элемент управления загружается на панель элементов управления в одной группе со строкой состояния.

Основными свойствами этого элемента управления являются:

Max и ***Min*** – для установки диапазона изменения контролируемого параметра. Если необходимо выводить результат в процентах, то свойству Min надо присвоить значение 0, а свойству Max – 100;

Value – возвращает текущее значение. Определение значения этого свойства пре-

доставляется разработчику, так как сам элемент управления не имеет возможности отслеживать продвижение процесса:

```
'пример для загрузки нескольких файлов
Sub LoadFiles()
    Progress1.Min=0
    Progress1.Max=nFiles
    For i=1 To nFiles
        Call LoadFile(i) ' Вызов процедуры загрузки файлов
        Progress1.Value=i
    Next i
End Sub
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначена строка состояния?
2. Как создать строку состояния?
3. Каким образом осуществляется настройка панелей строки состояния?
4. Каково назначение элемента управления ImageList?
5. Как создать список изображений на этапе разработки программы?
6. Как добавить новые изображения в список в процессе выполнения программы?
7. Поясните порядок создания панели инструментов.
8. Как создать кнопку для панели инструментов?
9. Для чего предназначен индикатор процесса?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с рядом элементов управления операционной системы Windows, позволяющих создавать приложения, не отличающиеся от стандартных приложений Windows.

Строка состояния (StatusBar) позволяет выводить оперативную информацию о состоянии приложения: дату, время, состояние нажатых функциональных клавиш, значения контролируемых параметров.

Список изображений (ImageList) хранит изображения рисунков, которые могут использоваться другими приложениями.

Элемент управления ToolBar позволяет создавать стандартную панель инструментов, которая облегчает управление программой, создает привычную среду для пользователя.

4.8. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ ДЛЯ РАБОТЫ С ВНЕШНИМИ УСТРОЙСТВАМИ

4.8.1. СПИСКИ УСТРОЙСТВ, КАТАЛОГОВ И ФАЙЛОВ

Visual Basic 6.0 имеет три стандартных элемента управления, предназначенные для работы с файлами и каталогами: ***DriveListBox*** (Список устройств), ***DirectoryListBox*** (Список каталогов), ***FileListBox*** (Список файлов). Все эти элементы используются, как правило, совместно. Взаимодействие между ними осуществляется через событие *Change*. Большинство их свойств совпадают со свойствами поля со списком.

Список устройств

Этот элемент управления предназначен для отображения списка всех доступных дисков и устройств системы и обеспечивает возможность их выбора.

Основным свойством элемента *DriveBox* является свойство ***Drive***, которое возвращает выбранный диск или устройство, например, "C:\".

Список каталогов

Данный элемент управления предназначен для выбора каталогов. Он отображает структуру каталогов выбранного диска и позволяет осуществлять выбор и смену каталогов. Основным свойством списка каталогов является свойство ***Path***. Оно возвращает полный путь к выбранному каталогу, включая имя диска (например, C:\Windows\Word).

Список файлов

Список файлов отображает файлы текущего каталога и обеспечивает их выбор. Основными свойствами данного элемента управления являются свойства *FileName* и *Pattern*.

Свойство ***FileName*** содержит имя выбранного файла (например, Bock.doc).

Свойство ***Pattern*** позволяет установить типы файлов, которые должны отображаться в списке. Для установки типов файлов допускается использование маски: *.ICO, *.BMP и т.д.

Например:

```
File. Pattern="*.ICO;*.BMP"
```

Здесь *File* – имя элемента управления *FileListBox*. Для организации совместного использования списков необходимо написать следующие тексты программ:

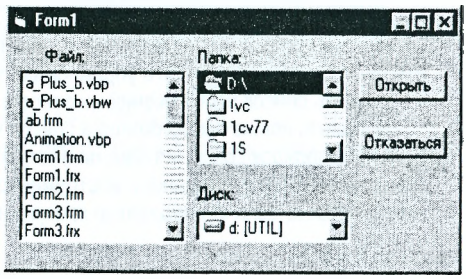


Рис. 4.43. Окно диалога для открытия файла

а) для списка устройств:

```
Private Sub Drive1_Change( )
   Dir1.Path=Drive1.Drive
End Sub
```

б) для списка каталогов:

```
Private Sub Dir1_Change( )
   File1.Path=Dir1.Path
End Sub
```

в) для списка файлов:

- для отображение полного маршрута выбранного файла напишем команду:

```
lblPath.Caption=File1.Path & "\" & File1.FileName.
```

Чтобы избежать отображения в маршруте излишнего количества символов "\" рекомендуется применять следующий код:

```
Private Sub File1_Click ( )
   If Right (File1.Path,1)= "\" Then
      lblPath.Caption=File1.Path & File1.FileName
   Else
      lblPath.Caption=File1.Path & "\" & File1. FileName.
   End If
End Sub
```

Пример использования данных списков приведен на рис. 4.43.

Рассмотренные элементы управления позволяют создать, при необходимости, средства интерфейса для открытия файлов и их сохранения. Однако для этих целей VB6 имеет более мощные средства. Это стандартные диалоговые окна Windows.

4.8.2. СТАНДАРТНЫЕ ОКНА ДИАЛОГА WINDOWS

Стандартные диалоговые окна представлены элементом управления **CommonDialog**. Загрузка этого элемента управления осуществляется командой ***Projec, Components.*** После входа в окно диалога установите флажок ***Microsoft Common Dialog. Control 6.0*** и щелкните клавишу OK.

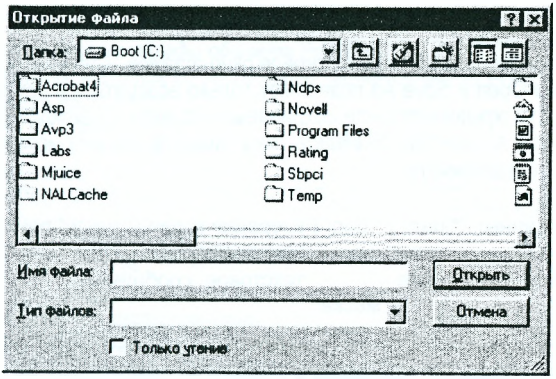


Рис.4.44. Стандартное окно диалога Open

Элемент управления Common Dialog, помещенный на форму, позволяет получить доступ к нескольким диалоговым окнам:

Open (Открыть) (рис.4.44). Это окно позволяет реализовать функцию открытия файла;

Save As (Сохранить как...), реализует функцию сохранения файлов на диске;

Print (Печать), реализует стандартную функцию Windows по настройке принтера печати документов;

Font (Выбор шрифта), позволяет выбрать тип шрифта и все его атрибуты;

Color (Цвет), предназначено для выбора одного из цветов стандартной палитры либо для создания нового цвета;

Help (Справочная система), позволяет создать интерфейс для работы пользователя со справочной системой Windows.

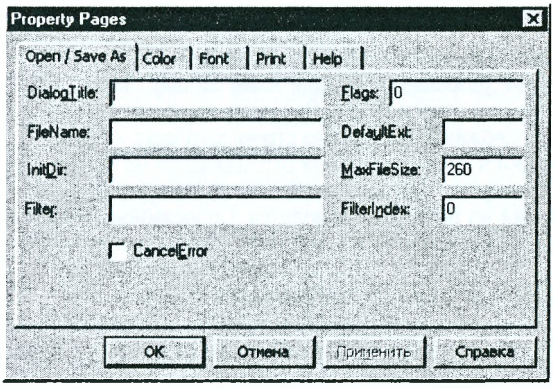


Рис. 4.45. Окно настройки свойств окон диалога

Диалоговые окна *Open* и *Save As* выглядят совершенно одинаково (рис.4.44) и похожи на стандартные окна диалога Windows.

Настройку окна диалога удобно вести с помощью окна свойств (рис. 4.45), которое вызывается с помощью контекстного меню объекта CommonDialog. Щелкните правой кнопкой мыши по окну объекта CommonDialog и выберите команду свойства – ***Properties***, откроется окно свойств ***Property Pages.*** Откройте за-

кладку ***Open / Save As.*** В поле ввода *DialogTitle* внесите название окна диалога "Открытие файла"; в поле ввода *InitDir* – внесите имя каталога, используемого по умолчанию; в поле ввода *Filter* – маску для чтения файлов с нужным расширением имени файла. Фильтр состоит из двух частей. Первая часть – текст, выводимый в строку ввода "тип файлов", например, "Файлы данных(*.dan)". Вторая часть – собственно фильтр – "*.dan", отделенный от первой части вертикальной чертой (рис.4.46). Если в фильтре используется несколько масок, то они имеют одинаковую структуру и отделяются друг от друга также вертикальной чертой.

Флажок *Flags* позволяет ввести в диалоговое окно флажок "Только для чтения".

Диалоговые окна Open и Save As позволяют только возвратить в программу полный путь (спецификацию) открываемого или сохраняемого файла. Спецификация выбранного файла сохраняется в свойстве ***FileName*** окна диалога. Задача использования этих данных ложится на программиста.

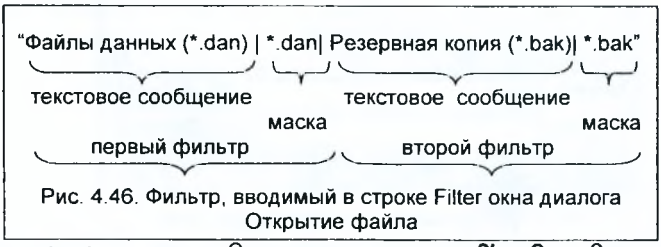


Рис. 4.46. Фильтр, вводимый в строке Filter окна диалога Открытие файла

Для вызова диалогового окна Open используется метод ***ShowOpen***. Синтаксис команды:

ИмяФормы.ShowOpen.

<переменная>= ИмяФормы.FileName.

Для вызова диалогового окна сохранения файла используется метод ***ShowSave***:

ИмяФормы. ShowSave.

<переменная>=ИмяФормы.FileName.

Здесь <переменная> – имя переменной для хранения и последующего использования имени открываемого или сохраняемого файла.

4.8.3. ПЕЧАТЬ ДОКУМЕНТОВ

Для управления выводом информации на печать используются метод ***PrintForm*** и объект ***Printer***.

Метод PrintForm

С помощью метода PrintForm на принтер выводится форма в виде растрового изображения с установленным в системе разрешением (чаще всего 96 dpi). Метод PrintForm может использоваться только для печати форм. При этом на принтер, установленный по умолчанию, выводится содержимое формы без строки заголовка и рамки.

Все элементы управления выводятся на печать так, как отображаются на экране, т.е. с соответствующими надписями, границами, видами шрифтов и т.д. Невидимые во время выполнения элементы управления на печать не выводятся. Содержимое элемента управления PictureBox выводится на печать только в том случае, если значение свойства AutoRedraw равно True.

Фрагмент кода для печати формы:

```
Load frmForm1
frmForm1.lblOutput.Caption=<выводимый текст>
frmForm1.picOutput.Picture=LoadPicture("c:\bibl.bmp")
frmForm1.PrintForm
Unload frmForm1
```

Объект Printer

Объект Printer предназначен для вывода на печать текста и графики.

В отличие от метода PrintForm, объект Printer позволяет выводить документ на печать с разрешением, установленным для принтера, а не для экрана, благодаря чему можно достичь лучшего качества печати. Кроме того, этот объект можно использовать для печати многостраничных документов. Однако весь процесс печати необходимо программировать.

Основные свойства и методы объекта Printer

Наиболее важным методом объекта Printer является метод ***Print***, с помощью которого текст передается на принтер:

```
Printer. Print " Здравствуйте "
Printer. Print " Печать документа "
```

Вывод осуществляется с верхнего левого угла печатной страницы, с использованием текущих параметров объекта Printer. Для изменения вида шрифта используется свойство объекта Font:

```
Printer. Font. Name = " Times New Roman "
Printer. Font. Size = 12
Printer. Font. Underline = True
Printer. Print = "Здравствуй, читатель "
```

Для изменения единицы измерения служит свойство ***ScaleMode***. По умолчанию в качестве единицы измерения используется твип. При установке значения свойства ScaleMode можно использовать константы ***vbCentimeters, vbMillimeters*** или ***vbPixels*** – сантиметры, миллиметры, пиксели, соответственно.

Для позиционирования точки вывода используются свойства ***CurrentX*** и ***CurrentY***. CurrentY устанавливает расстояние от верхнего края печатаемой области, CurrentX – от левого края печатаемой области. Ширина и высота печатаемой области в условных единицах измерения устанавливается свойствами ***ScaleWidth*** и ***ScaleHeight***.

Ширина и высота строки устанавливается с помощью свойств ***TextHeight*** и ***TextWidth***.

При выводе текста можно использовать функцию ***Tab***, а также управляющие символы (;) и (,) так же как в операторе Print. Для печати графических объектов используются методы ***Pset***, ***Line***, и ***Circle*** объекта Printer:

```
Printer. Line (1,1) – (10,5)
```

Готовые графические изображения различных форматов можно выводить на печать с помощью метода ***PaintPicture*** (см. раздел 4.9.4):

```
Printer.PaintPicture Form1.Picture, 0, 0, 1500, 1000, 0, 0, 7000, 5000
```

После направления всех данных на печать с помощью объекта Printer готовая страница пока еще находится в оперативной памяти. Для запуска процесса печати этот объект должен получить сообщение о том, что формирование этой страницы завершено. Для этой цели предназначен метод ***NewPage***. После того, как все страницы сформируются, вызывается метод ***EndDoc***, который направляет сформированный документ на принтер.

Для прерывания печати используется метод ***KillDoc***.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните назначение элементов управления DriveListBox, DirectoryListBox, FileListBox.
2. Назовите основные свойства и события элементов управления DriveListBox, DirectoryListBox, FileListBox.
3. Как организовать совместное использование элементов управления DriveListBox, DirectoryListBox, FileListBox?

4. Каково назначение элемента управления CommonDialog?
5. Как используются диалоговые окна элемента управления CommonDialog?
6. Как осуществляется настройка окна диалога Save или Open?
7. Для чего предназначен метод PrintForm?
8. Для чего предназначен объект Printer?
9. Как осуществляется настройка параметров шрифта для вывода данных на печать?
10. Как осуществляется управление позиционированием точки вывода при печати?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с элементами управления, позволяющими выполнять различные операции с файлами данных: открытие файлов, сохранение файлов на диске, вывод на печать.

Предпочтительнее использовать для указанных целей стандартное окно диалога Windows CommonDialog.

Необходимо, однако, помнить, что задача написания программы для использования данных элементов управления лежит на пользователе.

4.9. ГРАФИЧЕСКИЕ СРЕДСТВА VISUAL BASIC

4.9.1. ГРАФИЧЕСКИЕ ОБЪЕКТЫ VISUAL BASIC

Экран. Метод Scale

Экран

В графическом режиме экран видеомонитора представляет собой набор точек, расположенных по строкам. Каждая точка на экране называется пикселем, число точек по горизонтали и вертикали определяет разрешающую способность экрана. Для работы в среде Windows разрешающая способность должна быть не менее 800x600 пикселей.

Для работы с графикой Visual Basic 6.0 имеет *графические объекты, графические элементы управления* и *графические методы.*

Графические объекты. К графическим объектам относятся форма (Form) и графическое окно (Picture Box). К этим объектам могут быть применены графические методы.

Графические элементы управления – позволяют помещать на графические объекты линии и геометрические фигуры. К ним относятся элементы управления ***Line*** и ***Shape***. Особо следует выделить элемент управления ***Image***. Он не является ни графическим объектом, ни графическим элементом управления, так как не позволяет применять графические методы, но может использоваться для вставки рисунков.

Графический метод – это метод, который позволяет изображать на объекте данного класса какой-нибудь геометрический элемент, например точку, линию, окружность и т.д. Графический метод ориентирован на абсолютную или относительную систему координат экрана дисплея.

Рис. 4.47. Экранная система координат

Абсолютная система координат ориентирована на верхний левый угол экрана с значениями x=0; y=0, то есть представляет собой IV квадрант прямоугольной декартовой системы координат (рис.4.47).

Основной единицей измерения в VB является твип. Твип = 1/1440 логического дюйма. Логический дюйм – это расстояние на форме, которое при печати на принтере будет равно 1 дюйму (1 дюйм = 2,5 см). Используя свойство *ScaleMode*,

можно перейти к другим единицам измерения:

1. Твип (по умолчанию);
2. Точка (72 на дюйм);
3. Пиксель (пиксель – одна точка на экране монитора, число пикселей определяется установленным разрешением экрана Windows);
4. Символ (12 точек в высоту и 20 в ширину);
5. Дюйм;
6. Миллиметр;
7. Сантиметр.

Form1.ScaleMode = 3 – установлена единица измерения *пиксель*.
Form1.ScaleMode = 7 - установлена единица измерения *сантиметр*.

Метод Scale

Для установки другого масштаба, пользовательского, используется метод *Scale*. Синтаксис метода:

[имяОбъекта]. Scale (x1,y1) – (x2,y2)

где x1, y1 – координаты верхнего левого угла экрана; x2, y2- координаты правого нижнего угла экрана.

При отрицательных значениях координат меняется ориентация объекта.

```
Scale(-5,-10)-(5,10)  ' исходная, экранная система координат
Scale(-5,10)-(5,-10)  ' изменена ориентация оси Y
```

Можно использовать и такой способ установки координат:

```
Object. ScaleLeft =xxxx  - левый угол (верхний)
Object. ScaleTop = xxxx  - верхний угол (левый)
```

Например:

```
Picture1. ScaleLeft =100  - левый угол (верхний)
Picture1. ScaleTop = 50  - верхний угол (левый)
```

Можно также использовать для установки пользовательской системы измерений значение свойств ScaleWidth, ScaleHeight:

ScaleWidth = 3200 - масштаб по ширине
ScaleHeight = 2000 - масштаб по высоте.

Если в качестве объекта используется форма, то имя объекта в графических методах можно не указывать.

Объект Screen

Экран в VB представляет собой системный объект ***Screen***. Наряду с объектом Screen существует и свойство ***Screen*** объекта ***Global***.

Поскольку имеется только один единственный экран Windows, переменная типа Screen не объявляется, а используется системный объект Screen.

Объект Screen имеет ряд свойств:

ActiveControl - определяет активный элемент управления;
ActiveForm - определяет активную форму;
FontCount - количество доступных шрифтов;
Fonts( ) - возвращает имена всех доступных шрифтов;
Height - высота экрана;
MouseIcon - позволяет установить пользовательскую пиктограмму для курсора мыши;
TwipsPerPixelX - количество твипов в пикселе по горизонтали (разрешение по горизонтали);

TwipsPerPixelY - количество твипов в пикселе по вертикали (разрешение по вертикали);
Width - ширина экрана.

Параметры экрана по умолчанию измеряются в твипах:

```
HeightInTwips = Screen.Height ' высота экрана в твипах
WidthInTwips = Screen.Width ' ширина экрана в твипах
```

Эти параметры можно пересчитать и в пиксели:

```
HeigtInPixel = Screen.Height / Screen.TwipsPerPixelY ' высота экрана в пикселях
WidthInPixel = Screen.Width / Screen.TwipsPerPixelX ' ширина экрана в пикселях
```

Свойство ***ActiveControl*** позволяет обращаться к объектам без указания конкретного объекта. Чтобы обратиться к активному элементу, нужно написать текст программы следующего вида

```
Screen.ActiveControl.Свойство
```

Например:

```
Private Sub mnuDelete_Click ( )
    'Удаление выделенного текста.
    Screen. ActiveControl.SelText =""
End Sub.
```

Аналогично обращаются и к активной форме:

```
Screen.ActiveForm.ActiveControl.SelText = ""
```

Ключевое слово ***TypeOf*** позволяет проверить тип активного элемента управления:

```
Private Sub mnuDelete_Click()
    If TypeOf Screen.ActiveControl Is TextBox Then
        'Удаляется текст.
        Screen.ActiveControl.SelText = ""
    ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
        'Удаляется рисунок
        Screen.ActiveControl.Picture = LoadPicture()
    End If
End Sub
```

В данной процедуре делается проверка: если активный элемент *TextBox*, то удаляется выделенный текст, а если активным является элемент *PictureBox*, то удаляется рисунок (удаление рисунка осуществляется путем загрузки "пустого" рисунка – LoadPicture ()).

В приведенном примере текст программы включен в обработчик события пункта меню. Пункт меню не имеет фокуса. Нельзя использовать для удаления текста кнопку, так как она имеет фокус. Поэтому при щелчке мышью по кнопке она получает фокус и программа будет пытаться удалить текст с кнопки, но так как кнопка не имеет свойства SelText, то будет выдано сообщение об ошибке.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое графический объект? Назовите графические объекты.
2. Что такое графический метод? Назовите графические методы.
3. Какие графические элементы управления Вам известны?
4. Какие единицы измерения применяются в графических объектах?
5. Как пересчитать размеры экрана, заданные в твипах, в пиксели?
6. Расскажите назначение метода Scale. Приведите синтаксис метода.
7. Что такое объект Screen и каковы его основные свойства?
8. Для каких целей используется свойство ActiveControl объекта Screen?

4.9.2. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ LINE И SHAPE

Элемент управления Line

Элемент управления Line позволяет рисовать линии различной толщины и стиля (рис. 4.48).

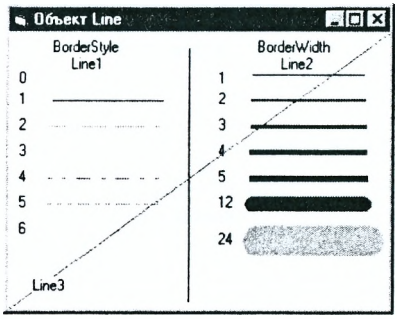


Рис.4.48. Свойства элемента управления Line

Этот элемент обладает 15 свойствами. Основными являются *X1, Y1, X2, Y2, BorderStyle , BorderWidth и BorderColor.*

X1, Y1, X2, Y2 – координаты концов линии, X1, Y1 – координаты левого конца линии; X2, Y2 - координаты правого конца линии.

BorderStyle - определяет стиль линии:

0 - невидимая; 1 – сплошная; 2 – пунктирная; 3 – пунктирная с коротким штрихом; 4 – штрихпунктирная; 5 – штрихштрихпунктирная; 6 –сплошная. Данное свойство работает только при значении свойства BorderWidth=1

BorderWidth - определяет толщину линии и может принимать любые значения кроме нуля.

BorderColor - определяет цвет объекта. Существует четыре способа задания цвета:

- непосредственное задание ***16-ричной константой***. Например: &H00000000& - черный цвет; &H000080FF& - красный цвет. Такой способ задания цвета в программе достаточно неудобен;
- использование ***RGB*** – функции: RGB (Red, Green, Blue).

RGB – функция формируется из трех цветов: красного, зеленого и синего. Каждый цвет задается числовой константой от 0 до 255. Эта функция позволяет формировать различные цвета и оттенки. Всего можно сформировать 255*255*255 различных оттенков. Например:

```
R=100: G=150: B=75
Line.BorderColor=RGB(R,G,B) 'темно-зеленый цвет
```

- использование ***констант Visual Basic.*** Имеется 8 констант: *vbBlack* - черный; *vbBlue* - синий; *vbCyan* - голубой; *vbGreen* - зеленый; *vbMagenta* - сиреневый; *vbRed* - красный; *vbWhite* - белый; *vbYellow* – желтый;
- использование функции ***QBColor (C)***, где С - номер цвета от 0 до 15:

Черный	- 0	Темно – серый	- 8
Темно-синий	- 1	Синий	- 9
Темно-зеленый	- 2	Зеленый	- 10
Темно-голубой	- 3	Голубой	- 11
Темно-красный	- 4	Красный	- 12
Темно-сиреневый	- 5	Сиреневый	- 13
Коричневый	- 6	Желтый	-14
Светло-серый	- 7	Белый	-15

Объект *Line* устанавливается на форму во время разработки программы, как и другие объекты управления. Положение объекта Line на форме можно изменить программным путем. Для управления объектом Line необходимо поместить объектный код в обработчик события ***Resize***.

Пример 4.28. Демонстрация свойств прямой линии (рис.4.46)

```
Option Explicit
Dim x1 As Single, x2 As Single, y1 As Single, y2 As Single
Dim i As Integer
-----------------------------------------------------------
Private Sub Form_Resize()
    ' горизонтальная черта по диагонали.
    Line3.x1 = 0
    Line3.y2 = 0
    Line3.x2 = Form1.ScaleWidth
    Line3.y1 = Form1.ScaleHeight
    Line3.BorderColor = vbRed
End Sub
-----------------------------------------------------------
Private Sub Form_Click()
    For i = 0 To 6
        Line1(i).BorderStyle = i
        Line1(i).BorderColor = QBColor(i + 8)
        Line2(i).BorderColor = QBColor(i + 1)
    Next i
End Sub
```

Объекты Line1 и Line2 представляют собой массивы элементов управления.

Элемент управления Shape

Элемент управления Shape служит для изображения геометрических фигур: квадратов, прямоугольников, эллипсов, окружностей.

Элемент Shape обладает практически теми же свойствами, что и элемент Line, но имеет и ряд специфических свойств. Основные свойства *Top, Left, Height, Width, Shape, BorderStyle, BorderWidth, FillStyle, FillColor.*

Top, Left, Height, Width – эти свойства аналогичны свойствам других элементов управления. Они определяют положение объекта на форме и его размеры.

Shape – определяет форму объекта и может принимать следующие значения: 0 - прямоугольник; 1 – квадрат; 2 – эллипс; 3 – круг; 4 – прямоугольник с закругленными углами; 5 – квадрат с закругленными углами (рис. 4.49.).

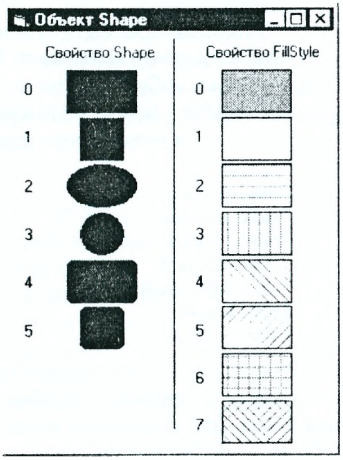


Рис.4.49. Свойства элемента управления Shape

FillStyle - обеспечивает автоматическое заполнение фигур, построенных с помощью графических методов. Это свойство имеет 8 значений: 0 - однотонное заполнение; 1 – пусто; 2- горизонтальные линии; 3- вертикальные линии; 4- диагонали верхние; 5- диагонали нижние; 6- сетка; 7- сетка диагональная.

BorderStyle, BorderWidth – определяют стиль контура и толщину линии соответственно. Эти свойства аналогичны соответсвующим свойствам объекта Line.

FillColor – определяет цвет заполнения объекта. Аналогичен свойству ***BorderColor*** объекта Line.

Пример 4.29. Демонстрация свойств элемента Shape.

Поместите на форму пять кнопок: Shape, FillStyle,

BorderStile, BorderWidth, FillColor и один элемент управления Shape, напишите в обработчиках событий Click кнопок объектные коды:

```
Option Explicit
Dim i As Integer, j As Integer
-----------------------------------
Private Sub cmdShape_Click()
    If i > 5 Then i = 0
    Shape1.Shape = i
    i = i + 1
End Sub
-----------------------------------
   Private Sub cmdBorderStyle_Click()
    If i > 6 Then i = 0
    Shape1.BorderWidth = 1
    Shape1.BorderStyle = i
    i = i + 1
End Sub
-----------------------------------
```

```
  Private Sub cmdBorderWidth_Click()
    If i = 0 Or i > 24 Then i = 1
    Shape1.BorderWidth = i
    i = i + 1
End Sub
-----------------------------------
Private Sub cmdFillColor_Click()
    If i > 15 Then i = 0
    Shape1.FillColor = QBColor(i)
    i = i + 1
End Sub
-----------------------------------
Private Sub cmdFillStyle_Click()
     If i > 7 Then i = 0
     Shape1.FillStyle = i
     i = i + 1
        End Sub
```

Для демонстрации свойств объекта Shape в данном примере необходимо щелкать мышью по соответствующей кнопке.

Управление пикселем

Чтобы использовать режим управления пикселем необходимо установить свойство ScaleMode =3

Число твипов, приходящихся на один пиксель, возвращают функции TwipsPerPixelX и TwipsPerPixelY.

Для управления цветом точки используется метод ***Pset***. Синтаксис метода:

Pset(x,y) [, C].

Метод Pset можно использовать для изображения графиков функций, а также для закраски фигур произвольной формы. К сожалению, в VB нет полезной функции для закраски фигур произвольной формы.

Для определения цвета точки используется метод ***Point***.

Этот метод возвращает длинное целое, используемое в кодировании (шестнадцатеричную константу). Синтаксис метода:

Object. Point(x,y)

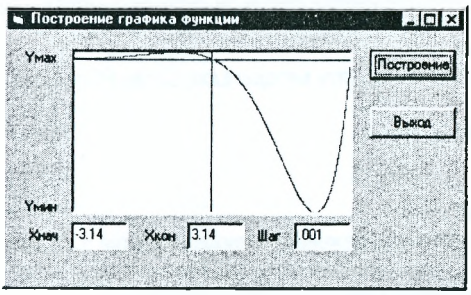


Рис.4.50. Построение графика функции

Пример 4.30. Построение графика функции (рис. 4.50).

Построить график функции e^x Sin(x) на отрезке [– π до π]

Порядок работы.

Для решения данной задачи необходимо протабулировать функцию на заданном отрезке с некоторым шагом и найти приближенное значение экстремумов функции. Затем провести масштабирование графического объекта (формы или элемента PictureBox) и построить график функции. Для построения графика функции повторите операцию табулирования функции и на каждом шаге стройте точку. Чтобы график функции был плотным, шаг табу-

лирования следует выбирать достаточно малым. С целью сокращения времени построения графика при некотором ухудшении качества для построения графика можно использовать метод Line (см. раздел 4.9.3).Для ввода данных используем окна ввода с именами: Хнач – Text1, Хкон – Text2 и шаг – Text3. Установим на форму объект PictureBox – Picture1.

```
Option Explicit
' объявление переменных
Dim x As Single, y As Single
Dim xn As Single, ymin As Single
Dim xk As Single, ymax As Single
Dim dx As Single
' функция пользователя
Function FNy(x As Single)
    y = Exp(x) * Sin(x)
    FNy = y
End Function
Private Sub Form_Click()
    Picture1.Cls
    Picture1.ScaleMode = 3
      xn = Val(Text1.Text)
      xk = Val(Text2.Text)
      dx = Val(Text3.Text)
  ' поиск максимума и минимума функции
     ymax = FNy(xn): ymin = ymax
     For x = xn To xk + dx / 2 Step dx
     y = FNy(x)
        If y > ymax Then ymax = y
        If y < ymin Then ymin = y
     Next x
' масштабирование объекта
     Picture1.Scale (xn, ymin)-(xk, ymax)
' построение графика функции
     For x = xn To xk + dx / 2 Step dx
             y = FNy(x)
             Picture1.PSet (x, y), vbRed
     Next x
End Sub
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите основные свойства элемента управления Line.
2. Перечислите основные свойства элемента управления Shape.
3. Какие способы используются в VB для задания цвета?
4. Приведите формат RGB – функции.
5. Какие графические методы используются для управления пикселем?
6. Приведите синтаксис метода PSet.
7. Напишите алгоритм построения графика функции с масштабированием.

4.9.3. ГРАФИЧЕСКИЕ МЕТОДЫ VISUAL BASIC

Visual Basic поддерживает несколько методов, которые можно использовать для рисования изображений в форме или элементе управления PictureBox. Эти методы можно использовать вместе с объектом Printer, чтобы направить вывод на принтер. Если метод вызывается без указания объекта, то его вывод направляется в форму, которая находится в фокусе в текущий момент времени.

Для создания графических образов можно применять методы *Line, Circle, PSet, Point, Paint Picture, Cls.*

Line – рисует линию или прямоугольник;

Circle - рисует окружность или овал;

PSet – размещает в указанном объекте точку;

Point – возвращает цвет определённой точки;

PaintPicture – рисует в указанном объекте изображение, которое хранится в другом элементе управления;

Cls- очищает область вывода указанного объекта;

Print – выводит текст в указанный объект.

Часть из этих методов можно использовать для изображения графических элементов в объекте. Некоторые свойства графических объектов влияют на работу этих методов. Эти свойства приведены в таблице 4.10.

Графический метод Line

Метод Line используется для рисования линий и прямоугольников.

Синтаксис метода:

[имя объекта.]Line [(x1,y1)] – (x2,y2), C, B[F]

Здесь так же как и в объекте Line *x1, y1*- координаты левого верхнего угла, *x2, y2*- координаты правого нижнего угла прямоугольника; *C* - цвет; *B* и *BF*- флаг. Флаг *B* определяет, что изображается прямоугольник, а не линия; флаг *BF* определяет, что изображается закрашенный прямоугольник. Если первый параметр опущен, то в качестве начальной координаты используется текущая точка.

```
Dim x1 As Single, x2 As Single, y1 As Single, y2 As Single.
x2= Me.ScaleWidth:  y2= Me.ScaleHeight
Line (0,0)- (x2,y2)
'рисуется линия из верхнего левого угла формы в правый
' нижний угол формы. Цвет линии принимается по умолчанию.
Line (100, 850) – (500,1800), vbBlue, B  'синий прямоугольник
Line (200,150) – (300,1000), vbRed, BF  'красный закрашенный
                                        ' прямоугольник
```

При изображении линии координаты начальной точки можно не указывать. В этом случае в качестве начальной точки используется текущая точка:

```
Current x= 1500 : Current y= 750
' установлены координаты начальной точки
Line – (x2,y2)  'рисуется линия из точки с координатами 1500, 750 _
            'в точку с координатами x2, y2 ( правый нижний угол)
```

В следующем примере показано влияние свойств формы на параметры графического объекта:

```
    ' Устанавливаем свойства  формы: стиль линии и толщину линии
    Me.DrawStyle = 2       'стиль линии - штриховая
    Me.DrawWidth = 1       ' толщина линии  1 твип
    Line (50,100) – (1200,500), vbGreen, B  'зелёный прямоугольник _
                ' вычерченный штриховой линией толщиной 1 твип
```

Пример 4.31. Построение графиков функций.

Построить на одном графике две функции и оси координат. Установить контроль полноты ввода данных. Для построения графиков использовать метод Line. Выполнить масштабирование графика.

Решение.

Форму разработаем согласно рис. 4. 50.

```
Option Explicit
Const Pi = 3.14159
-------------------------------------------------------------
Private Sub Form_Click()
    ' Объявление переменных
  Dim x As Single, y As Single, dx As Single
  Dim xn As Single, xk As Single
  Dim y1 As Single
    ' Контроль ввода данных.
 ' При ограниченном числе данных можно использовать простой
 ' способ контроля правильности ввода данных.
 If Text1.Text = "" Or Text2.Text = "" Or Text3.Text = "" Then
     MsgBox "Нет данных"
     Exit Sub
 End If
    ' Присвоение значений переменным
```

```
    xn = Val(Text1.Text)
    xk = Val(Text2.Text)
    dx = Val(Text3.Text)
        ' Масштабирование формы
    Scale (xn, 10)-(xk, - 5)
        ' Построение осей координат
    Line (xn, 0)-(xk, 0)
    Line (0, -5)-(0, 10)
        ' Построение графиков функций
    y = Exp(xn)  ' Вычисление значения первой функции в начальной точке
    PSet (xn, y), vbRed  'Построение начальной точки
    For x = xn To xk Step dx ' Построение графика функции
        y = Exp(x)
        If y <= 10 Then
            Line -(x, y), vbRed
        End If
    Next x
    y1 = Cos(xn)   'Вычисление значения второй функции в начальной точке
    PSet (xn, y1), vbBlue    'Построение начальной точки
    For x = xn To xk Step dx   'Построение графика функции
        y1 = Cos(x)
        Line -(x, y1), vbBlue
    Next x
End Sub
```

Метод Circle

Метод Circle используется для изображения эллипсов и окружностей.

Синтаксис метода:

[Объект.] Circle [Step] (x, y), R, C, [-] старт (start), [-] стоп (end), сжатие

Здесь *x*, *y* – координаты центра окружности; *R* – радиус, *C*- цвет; *старт* – начальный угол дуги, по умолчанию равен 0; *стоп* – конец дуги, по умолчанию равен 2π. Углы измеряются в радианах. Для указания угла в градусах необходимо использовать преобразование: $\alpha^*\pi/180$, где α- угол в градусах. Если перед значением угла стоит знак минус, то конец дуги соединяется с центром окружности.

```
Const PI As Double = 3.14159
' окружность красного цвета
Circle (200,200), 150, vbRed
'дуга эллипса в интервале от 30 до 120 градусов'
Circle (300,100), 100, vbGreen, 30/180*PI, 120/180*PI, 1.5
'сектор синего цвета
Circle (500,300), 200, vbBlue, -30/180*PI, -120/180*PI
```

Пример 4.32. Совместное использование свойств формы и свойств графических методов. Результаты управления изображениями на форме с помощью свойств формы приведены на рис. 4.51.

```
Option Explicit
Dim A As Single, An As Single, Ak As Single, N As Integer
Dim Pi As Single
'-------------------------------------------------
Private Sub Form_Click()
  Dim i As Integer, k As Integer
  Me.DrawStyle = 2      'стиль линии - штриховая
  Me.DrawWidth = 1      ' толщина линии  1 твип
```

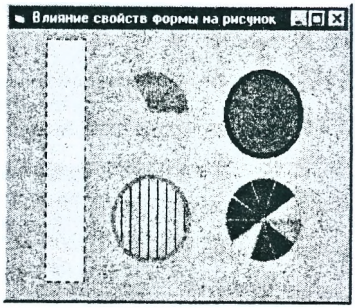


Рис. 4.51. Иллюстрация к примеру 4.32

```
    Me.FillColor = vbYellow 'цвет заливки желтый
    ' красный прямоугольник, закрашен желтым цветом
    Line (500, 100)-(1000, 3000), vbRed, B
    'сектор без окантовки, цвет сектора в операторе не указан
    'вычерченный штриховой линией толщиной 1 твип
     Me.DrawStyle = 5    ' нет линии
     Me.FillColor = vbGreen ' цвет зеленый
     Me.FillStyle = vbSolid ' сплошная заливка
     Me.Circle (1900, 1000), 500, , -0.1 * Pi / 180, -120 * Pi / 180
     'цвет окружности задан явно
     Me.DrawWidth = 2       ' толщина линии  2 твип
     Me.DrawStyle = 0       ' сплошная линия
     Me.FillStyle = vbSolid ' сплошная заливка
     Me.DrawWidth = 4       ' толщина линии  4 твипа
     Me.FillColor = vbMagenta 'цвет малиновый
     Circle (3400, 1000), 500, vbBlue
     ' круг с заливкой
     Me.ForeColor = vbGreen  'цвет контура зеленый
     Me.FillStyle = 3        ' заполнение вертикальная штриховка
     Me.FillColor = vbRed    ' цвет штриховки красный
     Circle (1900, 2250), 500
     ' Круг разбит на N секторов, число секторов задается
     ' в обработчике событий Load
     A = 360 / N
     Me.DrawWidth = 1     ' толщина линии  1 твип
     For i = 1 To N
       Ak = An + A
       Me.FillStyle = vbSolid
        If k > 15 Then k = 0
       Me.FillColor = QBColor(k)
       Circle (3400, 2250), 500, , -An / 180 * Pi, -Ak / 180 * Pi
       An = Ak: k=k+1
     Next i
End Sub
-------------------------------------------------------------------
Private Sub Form_Load()
   N = 10
   Pi = 4 * Atn(1)
End Sub
```

Построение столбиковой диаграммы

Пусть дано N параметров и их значения. Требуется построить столбиковую диаграмму. Диаграмма должна занимать часть формы и быть расположена в центре нее (Рис. 4.52). Данные вводить с клавиатуры в режиме диалога.

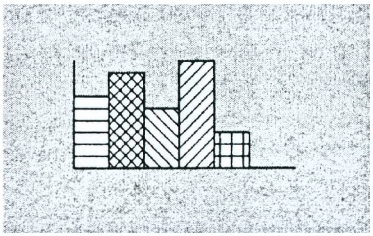


Рис.4.52 Столбиковая диаграмма

Порядок работы.

1. *Ввод данных.* Для ввода данных в режиме диалога воспользуемся оператором InputBox. Чтобы данные можно было удобно анализировать и использовать запишем их в массив, например A(N).

2. *Масштабирование.* Выберем ширину диаграммы произвольно, например *Bd=100*, а мас-

штаб по оси X примем равным удвоенному значению ширины диаграммы *Mx= 2*Bd =200*. Высоту диаграммы примем равной максимальному значению параметров *Hd= max*. Масштаб по оси Y примем равным удвоенному значению максимального параметра – *My=2*Hd*. Тогда масштаб формы определится оператором:

```
Scale (0, My)-(Mx, 0)
```

3. *Построение осей координат*. Начальное смещение по оси X примем равным половине разности ширины формы и ширины диаграммы – *cx = (Mx-Bd)/2*, Начальное смещение по оси Y примем равным половине разности высоты формы и высоты диаграммы – *cy = (My – Hd)/2*.

4. *Определение ширины столбца*. Чтобы все столбики диаграммы были одинаковой ширины, определим шаг *dx= Bd/(N+1)*, где *N* – число параметров. Число отрезков увеличено на единицу, чтобы ось X не сливалась с диаграммой и в то же время вся диаграмма располагалась в центре формы.

Для построения осей координат используем метод Line:

```
Line (cx, cy)-(cx, cy+Hd), vbBlue  'вертикальная ось
Line (cx, cy)-(cx + Bd, cy), vbBlue 'горизонтальная ось
```

5. *Построение диаграммы*. Для построения диаграммы применим цикл, так как значения параметров хранятся в массиве. Строить столбики диаграммы будем с помощью метода Line

```
Line (cx + (i - 1) * dx, cy)-(cx + i * dx, cy + A(i)), QBColor(i), BF
```

На форме не предусмотрено кнопок. Поэтому для запуска программы будем использовать событие Click формы. При запуске программы перед пользователем будет пустая форма. Чтобы пользователь не гадал, что надо делать дальше, поместим на форму надпись с текстом "Для начала работы щелкните мышью по форме". Когда пользователь щелкнет мышью по форме, уберем надпись, сделав ее невидимой.

Программа содержит значительное число переменных, без которых можно было бы обойтись при построении простой диаграммы. Однако, чтобы программа была универсальной, все переменные необходимы. Теперь, при необходимости можно дополнительно разработать форму для настройки параметров диаграммы. Ниже приведен текст программы.

6. *Текст программы:*

```
Option Explicit
Option Base 1  ' нумерация элементов массива наинается с единицы
Dim A() As Single
-------------------------------------------------------------------
Private Sub Form_Click()
    Dim i As Integer, N As Integer, S As Single, Bd As Single, Hd  As Single
    Dim max  As Single, dx As Single, Mx  As Single, My As Single
    Dim cx  As Single, cy As Single
    Label1.Visible = False
    max = -1E+38
    N = Val(InputBox("Укажите число параметров"))
    ReDim A(N) As Single
    max = -1E+38
    Bd = 100: Mx = 2 * Bd
    For i = 1 To N
        A(i) = Val(InputBox("Введите значение " & Str$(i) & "параметра"))
        If A(i) > max Then max = A(i)
    Next i
    Hd = max: My = 2 * Hd
    Me.Scale (0, My)-(Mx, 0)
```

```
    cx = Bd / 2: cy = Hd / 2
    dx = Bd / (N + 1)
    'Рисуем оси координат'
    Line (cx, cy)-(cx, cy + Hd), vbBlue ' вертикальная ось
    Line (cx, cy)-(cx + Bd, cy), vbBlue 'горизонтальная ось
    For i = 1 To N
        Line (cx + (i - 1) * dx, cy)-(cx + i * dx, cy + A(i)), QBColor(i), BF
    Next I
End Sub
```

Построение секторной диаграммы

Дано число параметров и их значения. Требуется построить секторную диаграмму. Ввод данных обеспечить в режиме диалога.

Порядок работы

1. *Ввод данных.* Для ввода данных в режиме диалога воспользуемся оператором InputBox. Чтобы данные можно было удобно анализировать и использовать запишем их в массив B(N). При вводе данных подсчитаем и сумму значений параметров - S. N – число секторов.

2. *Масштабирование.* Определим угол, приходящийся на единицу значения параметра *Da =360/S.*

3. *Построение секторов.* Построение секторов будем осуществлять с помощью цикла. Присвоим начальному углу *f1* значение нуль. Угол текущего сектора определим по формуле *f2= B(i)*da.*. Конечное значение угла сектора определим по формуле f3=f1+f2. Цвет секторов будем задавать с помощью свойства FillColor формы и функции QBColor(k). Чтобы программа работала корректно при любом числе секторов, необходимо контролировать параметр *k*. Значение параметра K не должно быть больше 15.

4. Текст программы.

```
Option Explicit
Const Pi As Single = 3.14159
Dim B() As Single
'-------------------------------------------------------------
Private Sub Form_Click()
    Dim n As Integer, S As Single, Da As Single
    Dim f1 As Single, f2 As Single, f3 As Single
    Dim R As Single, i As Integer, k As Integer
    Dim x As Single, y As Single ' x и y - координаты центра
    Cls
    n = Val(InputBox("укажите число секторов"))
    R = Val(InputBox("Укажите значение радиуса круга"))
    ' масштабирование формы
    Me.Scale (0, 0)-(4 * R, 4 * R)
    x = 2 * R: y = 2 * R     ' поместим круг в центре формы
    ReDim B(n)
    ' Ввод данных и вычисление суммы
    S = 0
    For i = 1 To n
        B(i) = Val(InputBox("Введите значение " & Str$(i) & "параметра"))
        S = S + B(i)
    Next i
    Da = 360 / S
    k = 0: f1 = 0
    ' Рисование секторов
    FillStyle = vbSolid ' сплошное закрашивание
```

```
    For i = 1 To n
      f2 = B(i) * Da
      f3 = f1 + f2
      If k > 15 Then k = 0
      FillColor = QBColor(k) ' цвет сектора
      Circle (x, y), R, QBColor(k), -f1 / 180 * Pi, -f3 / 180 * Pi
      k = k + 1
      f1 = f3
    Next i
End Sub
```

Метод Print

Метод Print "рисует" текст на объекте, как и другие графические методы. Его можно использовать в сочетании с другими методами для создания диаграмм или для анкетирования существующих растровых изображений. Синтаксис метода

```
Print "текст"
```

Работа метода Print зависит от значений некоторых свойств объекта, в который выводится текст:

CurrentX – горизонтальная координата точки вывода;

CurrentY – вертикальная координата точки вывода;

Font - тип и размер шрифта;

ForeColor - цвет текста;

FontTransparent – определяет, будет ли просвечиваться фон формы сквозь незаполненные части текста.

Например:

```
CurrentX=100
CurrentY=100
'Тип шрифта, начертание, высота и некоторые атрибуты могут
' выбираться в режиме диалога,  параметры объекта Font
' могут устанавливаться также программным путем
Me.Font.Italic = True   'курсив
Me.Font.Size = 12
Me.ForeColor=vbRed
Print "Ввод текста красным цветом"
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные графические методы.
2. Приведите синтаксис метода Line. Дайте пояснения.
3. Приведите синтаксис метода Circle. Поясните назначение опций.
4. Напишите алгоритм построения графика функции.
5. Напишите алгоритм построения столбиковой диаграммы.
6. Напишите алгоритм построения круговой диаграммы.
7. Какие свойства графических объектов влияют на работу графических методов?
8. Приведите синтаксис метода Print. Какие свойства графического объекта используются совместно с методом Print?

ЗАКЛЮЧЕНИЕ

Графические объекты позволяют вставлять рисунки, картинки, к ним могут быть применены графические методы. Visual Basic имеет два таких графических объекта: форму и рисунок. Еще один элемент – Картинка также позволяет вставлять рисунки, но не обладает свойствами и методами для вставки графических объектов.

Visual Basic имеет два графических элемента управления: Shape и Line, которые позволяют изображать на графических объектах такие графические примитивы, как линии, прямоугольники, овалы.

К графическим объектам могут быть применены графические методы, которые позволяют выполнять те же функции, что и графические элементы, но имеют значительно больше свойств. Некоторые свойства графических методов зависят от значений свойств графических объектов.

4.9.4. ОБЪЕКТЫ PICTUREBOX, IMAGE

Понятие о векторной и растровой графике

Растровые изображения представляют собой рисунки, состоящие из пикселей - точек на экране компьютера, формирующих картинку.

В векторной графике, фигуры представляются последовательностью точек, соединенных отрезками линий и кривыми.

Эти особенности определяют достоинства и недостатки каждого из этих методов представления рисунков. Растровые изображения позволяют получать рисунки высокого качества при постоянном разрешении. При изменении разрешения экрана или изменении размеров рисунка качество рисунка ухудшается. При увеличении размера растрового изображения появляется ступенчатость за счет увеличения размеров точки на экране, при уменьшении размеров рисунка качество рисунка также ухудшается за счет исключения отдельных пикселей. Растровые изображения требуют для своего хранения большего объема памяти, так как для каждого пикселя необходимо хранить значение цвета.

Таблица 4.8 Типы файлов, загружаемых в объект PictureBox

Расширение имени файла	*Тип файла*
.bmp	побитовый (то есть несжатый) растровый файл[9]
.dib	устройство-независимый растровый файл (Device Independent Bitmap)
.ICO, .CUR	пиктограммы, значки объектов и формы курсора, соответственно
.wmf	метафайл Windows – векторное изображение
.emf	расширенный метафайл
.gif	Graphics Interchange Format - растровый формат широко используемый на Web - страницах в Internet[10]
.jpg, .peg	Joint Photographic Experts Group – растровый формат, для уменьшения размера файла используется метод сжатия с потерей качества, широко используется на Web-страницах в Internet[11]

Графические элементы в векторном файле называются объектами. Каждый объект является самостоятельной единицей с такими свойствами, как цвет, форма, толщина линии (абрис), координаты начальной и конечной точек отрезка и размер. Поэтому размеры векторных графических файлов существенно меньше, чем размеры растровых файлов. Размеры векторных изображений легко изменяются без потери качества.

Visual Basic 6.0 имеет возможность работать с различными типами графических файлов (табл. 4.8).

---

[9] В стандарте Window для файлов .bmp предусмотрена возможность их сжатия, но только в рамках алгоритма RunLength, когда повторяющиеся байты (или биты) заменяются числом повторений и самим повторяющимся байтом.

[10] В файлах формата .gif используется сжатие по алгоритму Лемпела-Зива (LZW) без потери информации, эти файлы допускают использование простейших вариантов анимации в виде нескольких сменяющих друг друга картинок

[11] В файлах формата .jpg для сжатия используется быстрое преобразование Фурье для трех составляющих картинки, яркостной и двух цветоразностных составляющих, как в телевидении. При этом отбрасываются высшие составляющие разложений Фурье, что обеспечивает сжатие файла в 10 – 20 раз.

(Примечания по п. 10, 11 и 12 Колосова С. В.)

Окно с рисунком

Элемент управления PictureBox - Окно с рисунком - предназначен для отображения рисунков и других графических объектов. Этот элемент управления является элементом-контейнером, поэтому его можно использовать для объединения других элементов.

События объекта PictureBox обычно не обрабатываются.

Основные свойства объекта: *Align, Autosize, Picture.*

Align - определяет положение PictureBox в форме: 0 - None, 1 - Top, 2 - Bottom, 3 - Left, 4 - Right. В зависимости от значения этого свойства объект будет закрепляться у одного из краев формы или будет сохранять положение, заданное разработчиком (None). Если элемент управления закрепляется у одного из краев формы, то его размер (ширина и высота) всегда будут устанавливаться в соответствии с размерами формы.

Autosize - определяет, будут ли размеры элемента управления автоматически изменяться для отображения рисунков различного размера. Если свойство *Autosize* установлено в *False*, то в объект помещается весь рисунок, но в зависимости от его размера будет видна либо часть рисунка, либо будут оставаться пустые места на экране. Если свойство *Autosize* установлено в *True*, то размеры элемента управления автоматически изменяются до размеров рисунка.

Picture - позволяет загружать графические объекты и сохранять их, содержит отображаемый графический объект.

Загрузка рисунков

Для загрузки графического объекта щелкните мышью по кнопке *троеточие* в поле свойства *Picture* – появится диалоговое окно *Load Picture*, которое позволяет осуществить поиск и загрузку нужного файла с диска. Напимер: C:\Windows\облака.bmp

В процессе работы программы загрузку изображений можно осуществлять с помощью функции ***LoadPicture***:

```
Объект.Picture = LoadPicture("спецификация_файла")
```

Загружаемые во время разработки файлы хранятся вместе с формой, что увеличивает размер программы и время ее загрузки.

Сохранить изображение можно функцией ***SavePicture***. Файлы сохраняются в формате .BMP:

```
SavePicture(Picture1.Picture, "BUILD.BMP")
```

Методы объекта PictureBox позволяют нарисовать точку, линию и окружность, а также вывести текст (метод Print).

Выгрузка рисунков

Рисунки, помещенные в PictureBox, не удаляются, а выгружаются. Для выгрузки рисунка в процессе разработки программы необходимо выделить в строке свойства *Picture* записанное слово, например, *bitmap* и нажать клавишу ***Del***.

Выгрузка рисунка в процессе работы осуществляется также функцией LoadPicture(), но без указания имени файла:

```
Объект.Picture = LoadPicture()
```

Пример 4.33. Вывод текста и рисунка

```
Private Sub Form_Click()
Cls
Dim X As Single, Y As Single
picPicture1.CurrentX = 100
picPicture1.CurrentY = 100
picPicture1.Print "Ввод текста красным цветом"
```

```
'Тип шрифта, начертание, высота и некоторые атрибуты
' можно выбирать и  в режиме диалога
picPicture1.Font.Italic = False
picPicture1.Font.Size = 12
picPicture1.ForeColor = vbRed
picPicture1.CurrentX = 400
picPicture1.Print "Ввод текста красным цветом"
picPicture1.CurrentX = 700
picPicture1.CurrentY = 600
picPicture1.Font.Size = 8
picPicture1.ForeColor = vbBlack
picPicture1.Print "Ввод текста черным цветом, при X=700 твипов "
' Управление рисунком
X = 2000: Y = 1000
picPicture1.DrawWidth = 15
picPicture1.PSet (X, Y), vbRed
picPicture1.CurrentX = 900: picPicture1.CurrentY = 900
picPicture1.Print "Рисунок"
picPicture1.DrawWidth = 2
picPicture1.Line (0, 0)-(3000, 1500), vbBlue
picPicture1.Circle (2000, 1000), 500, vbGreen
picPicture1.CurrentX = 1100: picPicture1.CurrentY = 1200
picPicture1.Print "Рисунок1"
End Sub
```

Элемент PictureBox позволяет преобразовывать одни форматы изображений в другие. Например, пиктограмму (.ICO) можно преобразовать в растровый рисунок (.bmp). Для этого надо загрузить пиктограмму, а затем сохранить ее с расширением .bmp. Обратное преобразование не возможно.

Элемент управления Image

Элемент управления *Image* также создан для отображения рисунков. Но в отличие от *PictureBox*, он не является элементом - контейнером. Элемент управления *Image* не позволяет ни рисовать, ни группировать объекты. Однако Image использует меньше ресурсов и перерисовывает быстрее, чем PictureBox. Поэтому для отображения рисунков Image является более предпочтительным.

Основными свойствами элемента Image является свойство *Stretch* и *Picture*.

Свойство ***Picture*** аналогично соответствующему свойству элемента PictureBox.

Свойство ***Stretch*** позволяет устанавливать соответствие между размерами элемента управления и размером рисунка. Если свойству Stretch присвоено значение *True*, то размеры рисунка изменяются до размеров элемента управления Image, в противном случае элемент управления изменяется до размеров рисунка.

Загрузка изображений в форму

Изображения можно помещать непосредственно в форму. При этом изображение помещается от верхнего левого угла формы, никак не масштабируется и не растягивается.

Изображения не просвечиваются сквозь элементы управления за исключением Label и Shape. Чтобы элементы управления Label и Shape были прозрачны, значение их свойств *BackStyle* должно быть равно *Transparente*.

Основным преимуществом помещения изображения непосредственно в форму является то, что при этом используется меньше ресурсов системы, чем при предварительном размещении его в элементах управления PictureBox или Image.

Основные недостатки размещения изображений в форме:

- нельзя скрыть изображение, его можно только загрузить или выгрузить;
- нельзя управлять расположением изображения в форме;
- в форму одновременно можно поместить только одно изображение;
- размер изображения нельзя изменить. Оно помещается в форму в своем оригинальном размере;
- время на перерисовку формы требуется больше.

Метод Picture формы MDI позволяет поместить фоновое изображение на задний план.

Пример 4.34. Загрузка и выгрузка рисунков.

Установим на форму элементы управления Image и PictureBox. Свойство AutoSize объекта PictureBox и свойство Stretch объекта Image установим в True.

Загрузим в форму и элементы управления Image и PictureBox рисунки. Для этого необходимо выделить объект, щелкнуть по строке свойства Picture и выбрать в диалоговом окне нужный файл, например: C:\Windows\Лес.BMP, C:\Windows\Волны.BMP, C:\Windows\Облака.BMP. Напишем текст программы.

```
Private Sub Form_Click()
    Dim i As Double
    'выгрузка рисунков
    Form1.Picture = LoadPicture()
    Image1.Picture = LoadPicture()
    picPicture1.Picture = LoadPicture()
    For i = 1 To 50000: DoEvents: Next I  ' пауза
    ' Загрузка рисунка
    Form1.Picture = LoadPicture("C:\Windows\Наждак.BMP")
    Image1.Picture = LoadPicture("C:\Windows\Облака.BMP")
    picPicture1.Picture = LoadPicture("C:\Windows\Лес.BMP")
End Sub
```

Управление графическими объектами

Visual Basic имеет ряд свойств, позволяющих управлять перерисовкой графических объектов.

Свойство AutoRedraw

Свойство AutoRedraw графических объектов служит для их перерисовки. Если значение свойства равно False, то VB сохраняет копию объекта в памяти и перерисовку графического объекта надо будет производить самостоятельно. Для ускорения выполнения программы при использовании графики свойство AutoRedraw следует установить в True.

Существуют некоторые различия, как свойство AutoRedraw, будучи установлено в True, работает с формами и графическими окнами.

При уменьшении форм VB сохраняет копию всего экрана. Для увеличения формы этого не требуется, так как графическая информация не теряется. Опция True для сохранения всего экрана и быстрого выполнения требует большого объема памяти. Но если графический объект некорректно подогнан к форме, надо использовать эту опцию.

Для графических окон VB сохраняет изображение только по наибольшему размеру текущего окна, ничего нового не появляется, даже если окно было раньше увеличено. Поэтому рисование в графических окнах требует меньше памяти, чем рисование в форме, даже если графическое окно заполняет всю форму.

Свойство AutoRedraw определяет также, будут ли результат работы графических методов автоматически обновляться в окне, если оно скрыто (только для Form и Pic-

tureBox). При установке в True результаты работы графических методов будут автоматически обновляться в окне, даже если оно скрыто.

Событие ***Paint*** также служит для перерисовки объекта. События *Paint*, по сравнению с методом *AutuRedraw*, требует меньше памяти, но более медленно перерисовывает изображение.

> *В процедуре Paint нельзя размещать никаких инструкций по перемещению или изменению размеров объектов.*

Метод Refresh

Этот метод имеется у форм и элементов управления. Он приводит к немедленному обновлению формы или элемента управления и позволяет наблюдать подготовку изображения даже если свойство AutoRedraw равняется True. При использовании данного метода VB будет вызывать процедуру обработки событий Paint, созданную для данного объекта. Наиболее часто данный метод используется в процедуре ***Form_Resize,*** чтобы вывести повторно на экран любые графические изображения, подготовленные в Paint. Кроме того, поскольку VB управляет обновлением экрана в течение времени простоя, иногда требуется брать управление данными операциями на себя. В любом месте при использовании метода Refresh происходит немедленная перерисовка изображения объекта и генерация событий Paint, если данный объект имеет его.

Синтаксис использования метода:

```
Private Sub Form_Resize ( )
   Form1.Refresh
End Sub
```

При использовании метода Refresh изображение выводится на экран за несколько этапов. Однако каждый раз при этом VB рисует изображение по точкам. Такой способ занимает много времени.

Для перерисовки объекта необходимо периодически использовать метод Refresh.

> *Метод Refresh обычно используется в процедуре Form_Resize для отображения заново на экране монитора любой графики, которая обрабатывается в процедуре Paint.*

Свойство ClipControls

Свойство ClipControls влияет на скорость выполнения программы.

Если свойство ClipControls установлено в True (по умолчанию), а AutoRedraw в False, то VB перерисовывает весь объект. Если свойство ClipControls установлено в False, тогда Visual Basic перерисовывает только заново открываемую область.

При установке ClipControls в True вокруг неграфических элементов управления создается усеченная область. Это означает, что VB создает контур формы и элементов управления и хранит их в памяти. Установка этого свойства в False может уменьшить время, необходимое для рисования и перерисовки объекта, вследствие того, что усекаемая область создается в памяти. Чем сложнее объект, тем больше требуется времени на его обработку. Усеченные области исключают такие элементы управления, как Image, Label, Line и Shape.

Обобщенные сведения о влиянии свойства ClipControls и метода AutoRedraw на работу программы приведены в табл. 4.12.

> *Для увеличения скорости работы программы свойство AutoRedraw следует установить в True, а ClipControls в False.*

Таблица 4.12 Влияние свойств ClipControls и AutoRedraw на работу программы

ClipControl	*Autoredraw*	*Результат*
True (по умолчанию)	False	VB перерисовывает весь объект
False	False	VB перерисовывает заново только вновь открываемую форму
False	True	Увеличивает скорость работы программы

Метод PaintPicture

Метод PaintPicture перерисовывает изображение, находящееся в одном (исходном) объекте в другой. Задавая соответствующие значения аргументов *Height* и *Width* исходного и результирующего объектов, можно увеличить или уменьшить размер исходного изображения. Синтаксис данного метода:

ОбъектНазначения.PaintPicture ИсходныйОбъект.Paint X, Y, B, H, X1, Y1, [B1,H1]

Здесь *ОбъектНазначения* – имя объекта, куда будет помещаться изображение, *ИсходныйОбъект* – объект, откуда берется часть изображения; X, Y – координаты верхнего левого угла нового рисунка, B, H – ширина и высота результирующего рисунка; X1, Y1 – координаты рисунка в исходном объекте; B1, H1 – размеры исходного рисунка.

```
Picture1.PaintPicture Form1.Picture, 0, 0, 1500, 1000, 0, 0, 7000, 5000
```

В данном примере копируется рисунок из формы в элемент управления PictureBox с уменьшением.

Если размеры исходного объекта (B1, H1) не указаны, то копируется все изображение исходного рисунка.

Метод PaintPicture может применяться в следующих случаях:

- для выполнения масштабирования рисунка, чтобы можно было подробнее рассмотреть некоторую область изображения, например при предварительном просмотре;
- для копирования или затирания определенной области изображения;
- для перемещения содержимого элемента управления PictureBox в объект Printer, например, если нужно поместить рисунок в отчет.

Метод Point

Метод Point возвращает RGB – цвет определенной точки,

синтаксис: Point (x, y)

Пример 4.35. Определить цвета и принадлежности точки прямоугольнику.

```
Option Explicit
'-------------------------------------------------------------
Private Sub Form_MouseDown(Button As Integer, _
  Shift As Integer, X As Single, Y As Single)
  Dim Массив As Variant, Массив1 As Variant
  Dim i As Integer
  Me.AutoRedraw = True
  Массив = Array(vbBlack, vbBlue, vbCyan, vbGreen, _
       vbMagenta, vbRed, vbWhite, vbYellow)
  Массив1 = Array("черного", "синего", "голубого", "зеленого", _
       "малинового", "красного", "белого", "желтого")
  Line (100, 500)-(1500, 1500), vbRed, BF
  Line (1600, 500)-(3000, 1500), vbCyan, BF
  Line (100, 1600)-(1500, 3000), vbYellow, BF
  Line (1600, 1600)-(3000, 3000), vbWhite, BF
  If Not Point(X, Y) Then
     For i = 0 To 7
```

```
        If Point(X, Y) = Массив(i) Then
          MsgBox "Точка (" & X & " , " & Y & ") находится" & " " & _
            vbCr & "в прямоугольнике" & " " & Массив1(i) & " " & "цвета"
        End If
      Next i
    Else
      MsgBox " Точка находится вне прямоугольника"
    End If
  End Sub
```

Сохранение изображений

Visual Basic позволяет сохранить изображения, нарисованные в форме или графическом окне, например:

```
SavePicture   ИмяОбъекта.Image.ИмяФайла
```

Операционная система использует свойство *Image* для определения того, как нарисовано изображение: в форме или графическом окне. Если работа в графическом окне прекращается, то VB использует для сохранения текущую форму:

```
SavePicture   Image.ИмяФайла
```

Если изображение загружено первоначально из файла, определяющего свойство Picture, то VB сохраняет изображение в том же формате, что и оригинальный файл, иначе VB сохраняет файлы в формате побитового изображения - bitmap – файлы (.bmp).

Функция DoEvents

При организации вычислительных процедур с циклами Visual Basic по умолчанию не реагирует на события. Негативные последствия этого наглядно проявляются при работе с движущимися графическими объектами: объект не меняет положение или не изменяется предусмотренный цвет и т. д. Чтобы избежать таких последствий, необходим механизм отслеживания состояния операционной системы и реагирования на различного рода события.

Задачи такого рода выполняет функция ***DoEvents***. В каком бы месте программы ни стоял данный оператор, он сигнализирует Visual Basic о том, что управление передано операционной системе для обработки всех событий.

> *Функцию **DoEvents** нельзя использовать в процедуре обработки событий, которая вызывается несколько раз. Иначе можно организовать в программе бесконечный цикл.*

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем отличается растровое изображение от векторного?
2. Для чего предназначен графический объект PictureBox?
3. Какими свойствами обладает объект PictureBox?
4. Как влияет свойство AutoSize на размер рисунка, помещаемого в объект PictureBox?
5. Какие типы файлов можно загружать в VB?
6. Для чего предназначен элемент управления Image?
7. Поясните основные свойства элемента управления Image?
8. Какую функцию выполняет свойство AutoRedraw?
9. Для чего предназначен и как используется метод Refresh?
10. Как влияет свойство ClipControl на скорость выполнения программы?
11. Поясните назначение метода PaintPicture. Приведите синтаксис метода.
12. Поясните назначение метода Point.

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с понятиями растровой и векторной графики, а также с графическими объектами. Графические объекты Form, PictureBox позволяют использовть графические методы для изображения рисунков. Свойства графических методов могут зависеть от значений установленных свойств графического объекта. Рисунки можно вставлять непоредственно в форму. При этом обеспечивается экономия системных ресурсов, но существенно увеличивается время перерисовывания объекта.

Элемент управления Image не относится к графическим объектам, но позволяет вставлять рисунки. Объект Image перерисовывается быстрее, чем PictureBox, поэтому рисунки предпочтительнее размещать в нем.

Программы для управления графическими объектами целесообразно размещать в обработчике события Resize. Совместное использование свойств AutoRedraw и ClipControls позволяет увеличить скорость выполнения программы. Метод Refresh обеспечивает немедленную перерисовку объекта. Метод PaintPicture позволяет копировать графические объекты целиком или частями, а также выполнять их масштабирование. Полезно запомнить функцию DoEvents. Включение ее в любом месте программы обеспечивает передачу управления операционной системе при выполнении циклов.

4.9.5. АНИМАЦИЯ

Элемент управления Animation

Элемент управления Animation позволяет легко включать в программу различные анимационные эффекты. Он используется для воспроизведения не озвученных видеороликов в формате AVI. Видеоролик в данном формате фактически представляет собой набор растровых изображений (кадров). Кадры выводятся последовательно через равные промежутки времени, за счет чего и создается анимационный эффект, почти как в обычных мультфильмах. Обычно элемент управления Animation применяется там, где нужно "создать видимость работы", например, в процедуре копирования файлов в Windows.

Другим способом создания анимационных эффектов является использование таймера. Через определенные промежутки времени нужно просто изменять положение на экране элемента управления Image или другого рисованного объекта.

Настройка элемента управления Animation

Чтобы настроить элемент управления, его нужно сначала поместить в Форму. При этом создается контейнер для воспроизведения последовательности анимационных изображений. Чтобы теперь просмотреть фильм, нужно открыть конкретный файл и запустить его на воспроизведение. Для этой цели в форму надо поместить кнопку и в процедуру Click этой кнопки написать текст программы, который открывает и проигрывает видеофильм.

Элемент управления Animation имеет три метода: *Open*, *Play*, *Stop*.

Метод ***Open*** используется для открытия файла.

```
Animation1.Open "Спецификация файла"
```

Метод ***Play*** (или свойство ***AutoPlay*** элемента управления Animation) обеспечивает управление демонстрацией ролика. Этот метод имеет три параметра:

Repeat – определяет число повторных воспроизведений;

Start – определяет номер кадра, с которого должно начаться воспроизведение;

Stop – определяет номер кадра, на котором должно заканчиваться воспроизведение.

```
Animation1.Play  Repeat, Start, Stop
```

Метод ***Stop*** используется для остановки видеофильма.

Пример использования элемента управления Animation:

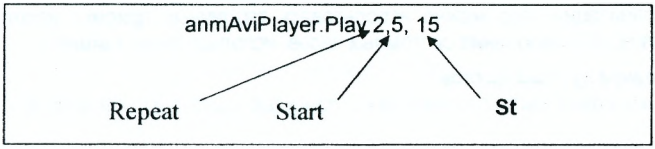


Рис. 4.53. Структура метода Play

```
anmAviPlayer.Open "C:\Progra~1\Micros~1\Common\Graphics\Video\ FileNuke.AVI"
anmAviPlayer.Play 2,5, 15
```

В приведенном примере, и на рис. 4.53, дважды воспроизводятся кадры с №5 по №15.

Создание анимации пользователем

Каждый пользователь при небольшом навыке может создать собственные анимационные эффекты. При этом можно использовать различные способы и возможности языка программирования:

- пересчет координат объекта и использование свойств *Top* и *Left* объекта или операторов *CurrentX*, *CurrentY* для переопределения координат объекта;
- использование метода ***Move***;
- использование буфера обмена ***Clipboard***;
- прямое присвоение значений свойств одного графического объекта другому.

Общий алгоритм работы программы при создании анимации следующий:

- воспроизвести изображение в начальной точке;
- сделать паузу на некоторое время, достаточное для фиксации изображения (десятые доли секунды). Длительность паузы, когда изображение на экране неподвижно определяет и скорость перемещения объекта;
- пересчитать координаты объекта;
- стереть изображение;
- воспроизвести изображение в новой позиции.

Самая большая проблема в анимации для VB состоит в необходимости перерисовки объекта каждый раз, когда он передвигается. Этот процесс занимает много времени. Рисование цветом фона тоже не работает, поскольку затирает все, что было до этого на экране монитора.

Ключом к решению данной проблемы является использование свойства DrawMode, определяющего как рисующий цвет взаимодействует с уже находящимися в объекте цветами.

Свойство DrawMode

Существует 15 возможных установок DrawMode. Во всех случаях VB сравнивает значение цвета пикселя на экране со значением цвета пикселя объекта, который рисуется на экране.

Если DrawMode = 7, то результатом его работы будет оператор ***Xor***. DrawMode = 6 – соответствует оператору ***Not***, а DrawMode = 4 определяет работу VB с оператором Not над значениями цветов переднего плана и использует значения этих цветов для рисования.

Если оператор Xor применяется дважды, то происходит восстановление первоначального цвета.

Повторное воспроизведение графического объекта при установленном значении свойства DrawMode 7 для формы или окна позволяет стереть построенное ранее изображение без потери другой информации.

Организация пауз

Для наблюдения процесса движения, особенно на быстродействующих компьюте-

рах, необходимо позаботиться о замедлении движения объекта. Это можно сделать двумя способами: уменьшением шага переноса и организацией пауз.

Для организации пауз можно использовать различные приемы: использование пустых циклов, использование системных часов, использование таймера.

Использование пустых циклов

Пустой цикл может быть организован с помощью одного из операторов цикла, например:

```
For i=1 To 1000: DoEvents: Next i
I=0: While I<1000: I=I+1: DoEvents: Wend
```

Недостатком использования пустых циклов для организации пауз является то, что длительность паузы будет зависеть от быстродействия компьютера.

Использование системных часов

Этого недостатка можно избежать, если использовать текущее время системных часов компьютера:

```
T=Time() или T=Timer()
While  Timer() – T < 2: DoEvents: Wend
```

Функция Time() возвращает текущее время в часах, минутах и секундах.

Функция Timere() возвращает время в миллисекундах.

Пример использования этих функций приведен на рис. 4.54.

Для использования функции Time() необходимо задать переменную типа Date. Имеется возможность определить время в минутах и секундах с помощью функций Minute(N) и Second(N).

Пример 4.36. Использование функций Time() и Timer() (рис.4.54)

```
Private Sub cmdStart1_Click()
  'Использвание функций Time() и Timer()
  Dim TT As Date, i As Integer
M:
  TT = Time()
  Text1.Text = TT
  Text2.Text = Minute(Time())
  Text3.Text = Second(Time())
  Text4.Text = Timer()
```

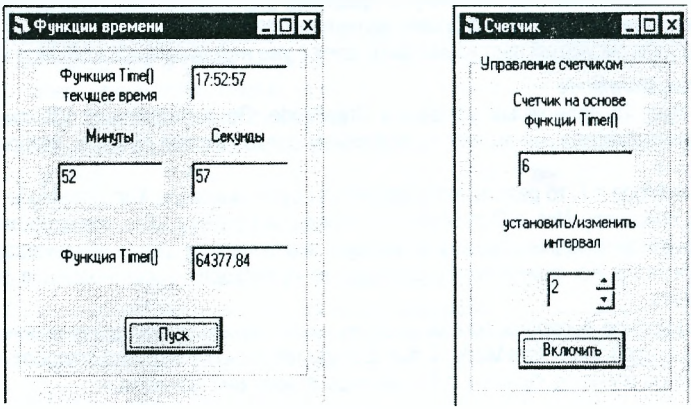


Рис. 4.54. Использование функций времени

```
    For i = 1 To 100: DoEvents: Next I 'пауза
    GoTo M
End Sub
```

Пример построения счетчика с использованием функции Timer()

```
Private Sub cmdStart_Click()
    Dim TT1 As Double, TT2 As Integer, dT As Integer
    TT2 = 0
    Text5.Text = TT2
m1:
    dT = Val(Text6.Text)      ' установка паузы в секундах
    TT1 = Timer()             ' запоминание текущего времени
    While Timer() - TT1 < dT
      DoEvents
    Wend
    TT2 = TT2 + 1
    Text5.Text = TT2
    GoTo m1
End
```

Использование таймера

Visual Basic позволяет устанавливать до 36 таймеров. Основные свойства таймера: *Intrval* и *Enabled*. Основное событие – *Timer*.

Interval - позволяет установить интервал выдачи сигнала от 0 до 10000 миллисекунд, что соответствует примерно одной минуте. Для получения больших интервалов времени необходимо применять счетчики.

Enabled – позволяет запускать и останавливать таймер. Если Enabled равно True, то таймер запускается. При установке значения свойства Enabled в False таймер останавливается.

Обработчик события ***Timer*** используется для размещения текста программы, которая должна выполняться через заданный интервал времени.

Пример 4.37. Использование элемента управления Timer.

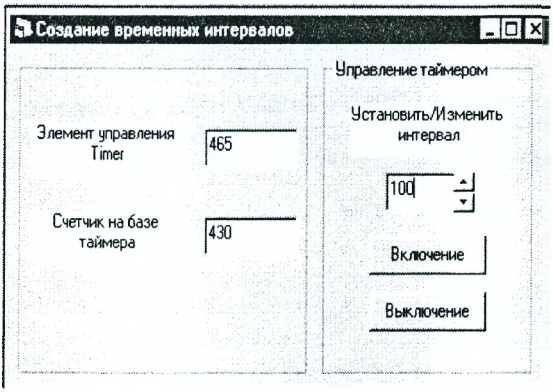


Рис. 4.55. Использование таймера

Установите на форму элементы управления согласно рис. 4.55, а также два таймера. Установите значения свойств Interval таймеров, отличные от нуля. На базе первого таймера создайте счетчик, который будет выдавать сигналы в соответствии с установленным интервалом. А на базе второго таймера создайте настраиваемый счетчик, который будет выдавать сигналы через пять интервалов таймера. Ниже приводится текст программы, соответствующий данной задаче.

```
Option Explicit
Dim TT2 As Double
---------------------------
Private Sub Form_Load()
   ' выключение таймеров
   Timer1.Enabled = False
   Timer2.Enabled = False
    Me.Height = 3720
    Me.Width = 6360
    TT2=0
End Sub
---------------------------
Private Sub Timer1_Timer()
   ' счетчик на базе таймера
   Static TT3 As Integer
   TT3 = TT3 + 1
   Text1.Text = TT3
End Sub
---------------------------
Private Sub Text3_Change()
'настройка интервала таймера
Timer2.Interval= Val(Text3.Text)
End Sub
---------------------------
```

```
Private Sub Timer2_Timer()
   Static TTEK As Integer
   TTEK = TTEK + 1
   If TTEK = 5 Then
      TT2 = TT2 + TTEK:
      Text2.Text = TT2
      TTEK = 0
   End If
End Sub
---------------------------
Private Sub cmdStart_Click()
   ' включение таймеров
   Timer1.Enabled = True
   Timer2.Enabled = True
End Sub
---------------------------
Private Sub cmdStop_Click()
   ' выключение таймеров
   Timer1.Enabled = False
   Timer2.Enabled = False
End Sub
---------------------------
Private Sub Form_Unload(Cancel As Integer)
    Unload Me
End Sub
```

Переменные TT3 и TTEK объявлены в процедурах как статические, чтобы при выходе из процедуры их значения не пропадали. Переменная TT2 объявлена переменной уровня формы и также позволяет накапливать значение счетчика.

В обработчике события Timer1_Timer() переменная TT3 будет изменять свое значение через интервалы, равные значению свойства Interval таймера1.

В обработчике события Timer2_Timer() переменная TTEK будет также изменять свое значение через интервалы, равные значению свойства Interval таймера2, а значения переменной TT2 будут меняться в пять раз реже. Таким способом можно задавать с помощью одного счетчика кратные интервалы времени.

Примеры анимации

Простая анимация

Пример 4.38. "Мяч" в клетке

Установите на форму элементы управления Shape и Timer. Установите требуемые значения свойств Shape, FillStyle и FillColor объекта Shape1. Установите требуемое значение свойства Interval таймера. И напишите приведенный ниже текст программы.

```
Option Explicit
Dim x As Single, y As Single, dx As Single, dy As Single
---------------------------------------------------------------------
Private Sub Form_Load()
' позиционирование "мяча" в центре формы
Shape1.Left = Me.ScaleWidth / 2 - Shape1.Width / 2
Shape1.Top = Me.ScaleHeight / 2 - Shape1.Height
 x = Shape1.Left: y = Shape1.Top
 dx = 100: dy = 100
End Sub
```

```
---------------------------------------------------
Private Sub Timer1_Timer()
  ' движение "мяча" вниз и вправо
  Shape1.Left = Shape1.Left + dx
  Shape1.Top = Shape1.Top + dy
     ' изменение направления движения мяча
  If Shape1.Top < 0 Then dy = dy * (-1)
  If Shape1.Top > Me.ScaleHeight - Shape1.Height Then dy = dy * (-1)
  If Shape1.Left < 0 Then dx = dx * (-1)
  If Shape1.Left > Me.ScaleWidth - Shape1.Width Then dx = dx * (-1)
End Sub
```

Скорость движения "мяча" можно регулировать, изменяя значение свойства Intreval таймера и значения переменных dx и dy.

Пример 4.39. Движение объекта по заданной траектории.

Написать программу движения круга по эллипсу.

Установите на форму таймер и запишите в обработчик события Timer1 приведенный текст программы. Свойству Interval таймера присвойте небольшое значение, например, 10. Установите свойству DrawMode значение Xor Pen (7), сохраните программу на диске и проверьте работу программы.

```
Option Explicit
Dim x As Single, y As Single
Dim a As Single, b As Single
Const pi = 3.14159
---------------------------------------------------
Private Sub Timer1_Timer()
 Dim i As Single, j As Single, t As Single
   a = 1000: b = 750
   For i = 0 To 2 * pi Step 0.01
      x = 2000 + a * Cos(i): y = 1500 + b * Sin(i)
      FillStyle = vbSolid ' сплошное закрашивание
      FillColor = vbRed  ' цвет красный
      Circle (x, y), 150, vbRed ' рисование объекта
      PSet (x, y), vbGreen
      For j = 1 To 100000:  Next j  ' пауза
      Circle (x, y), 150, vbRed      ' стирание объекта
   Next i
End Sub
```

Пример 4.40. "Броуновское движение".

В данном примере (рис. 4.56) объект движется по случайному закону. В качестве объекта можно использовать кнопку, метку, картинку и тому подобное. Регулировка скорости обеспечивается с помощью цикла While/Wend, в котором временной интервал задается с помощью системных часов. В программе используются следующие переменные:

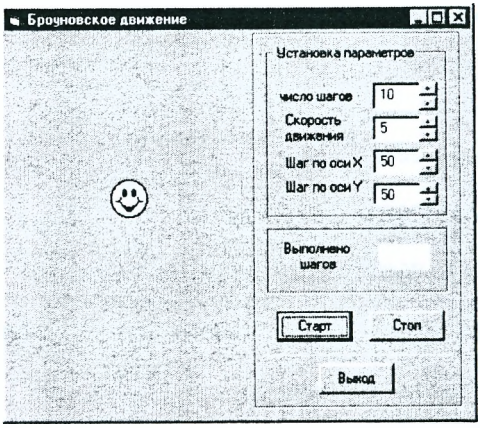


Рис. 4.56. Броуновское движение

TimerTimes – используется для контроля временного интервала;

NShagov – число шагов; dT – временной интервал. Чем больше временной интервал, тем реже выполняются шаги; dX – шаг по горизонтали; dY – шаг по вертикали; xmove, ymove – перемещение по горизонтали и по вертикали; x, y – текущие значения координат верхнего левого угла перемещаемого объекта. В качестве картинки загружен значок Face05 из папки Graphics, Icons, Misc.

Чтобы обеспечить прерывание цикла, введена логическая переменная Flag1. При нажатии кнопки Стоп переменной Flag1 присваивается значение истина – True и осуществляется выход из процедуры. Присвоение значений переменным dT, dX, dY осуществляется в момент ввода данных, для этого используются обработчики события Change полей ввода TextBox.

```
Option Explicit
Dim TimerTimes As Double
Dim NShagov As Integer, dT As Single
Dim dX As Single, dY As Single
Dim Flag1 As Boolean
---------------------------------------------------------------
Private Sub cmdExit_Click()
    End
End Sub
---------------------------------------------------------------
Private Sub cmdStart_Click()
    Dim x As Single, y As Single
    Dim xmove As Single, ymove As Single
    Dim i As Integer, j As Integer
    NShagov = Val(Text1.Text)
    Cls
    Flag1 = False
    Randomize (1)
    Me.ScaleMode = 3 ' шкала в пикселях
    Me.WindowState = 2
    x = Me.ScaleWidth / 2
    y = Me.ScaleHeight / 2
    Frame1.Left = Me.ScaleWidth - Frame1.Width
    PSet (x, y)
      For i = 1 To NShagov
      If Flag1 = True Then Exit Sub
      Label3.Caption = Str(i)
      If j > 15 Then j = 0
      xmove = dX * Rnd
      ymove = dY * Rnd
```

```
    If Rnd < 0.5 Then x = x + xmove Else x = x - xmove
    If Rnd < 0.5 Then y = y + ymove Else y = y – ymove
' проверка условий выхода картинки за границу поля
    If x < 0 Or y < 0 Or x > (ScaleWidth - Image1.Width) Or _
          y > (ScaleHeight - Image1.Height) Or _
         (x > (ScaleWidth - Frame1.Width - Image1.Width) _
         And y < (ScaleHeight - Frame1.Height - Image1.Height)) Then
              MsgBox "Я могу выйти за границу поля," & VbCr & _
                              "поэтому стою на месте"
    Else
          Image1.Move x, y
          Line -(x, y), QBColor(j)
    End If
    TimerTimes = Timer()
    While Timer() - TimerTimes < dT  ' временная задержка,
          DoEvents
    Wend
    j = j + 1
   Next i
End Sub
-------------------------------------------------------------------
Private Sub cmdStop_Click()
  Flag1 = True
End Sub
-------------------------------------------------------------------
Private Sub Form_Load()
   Me.Width = 6200
   Me.Height = 6100
   NShagov = 10
   Text1.Text = Str(NShagov)
   dX = 50: dY = 50: dT = 5
   Text2.Text = Str(dT):  Text3.Text = Str(dX):  Text4.Text = Str(dY)
End Sub
-------------------------------------------------------------------
Private Sub Text2_Change()
   dT = Val(Text1.Text) / 10
End Sub
-------------------------------------------------------------------
Private Sub Text3_Change()
  dX = Val(Text3.Text)
End Sub
-------------------------------------------------------------------
Private Sub Text4_Change()
  dY = Val(Text4.Text)
End Sub
```

В приведенном примере проводится сложная проверка условий выхода картинки за границы формы или попадание под панель управления:

```
    If x < 0 Or y < 0 Or x > (ScaleWidth - Image1.Width) Or _
          y > (ScaleHeight - Image1.Height) Or _
         (x > (ScaleWidth - Frame1.Width - Image1.Width) _
         And y < (ScaleHeight - Frame1.Height - Image1.Width)) Then
```

Проанализируем данное выражение.

x < 0 Or y < 0 - контроль выхода за левую или верхнюю границы формы;

x > (ScaleWidth - Image1.Width) - контроль выхода за правую границу формы;

y > (ScaleHeight - Image1.Height) - контроль выхода за нижнюю границу формы;

x > (ScaleWidth - Frame1.Width - Image1.Width) And y < (ScaleHeight - Frame1. Height - Image1.Height)) – контроль попадания картинки под панель управления, представленную рамкой Frame1.

Так как при перемещении объекта указываются координаты левого верхнего угла, то при контроле выхода картинки за нижнюю и правую границы области перемещения картинки Image1.учитываются также ее размеры.

Анимация посредством переноса изображений через буфер обмена

Независимо от способа создания рисунка, в VB предусмотрена возможность перенести его в другой элемент управления или другие приложения Windows. Для этого используются методы *SetData, GetData( ), GetForm, Clear.*

Метод ***SetData*** – перемещает данные в объект ***ClipBoard.***

Синтаксис метода:

Объект. SetData [данные],[формат]

Здесь:

Объект - буфер обмена, его идентификатор ClipBoard;

Данные – указывают, откуда переносятся данные (ImageBox, PictureBox и др.);

Формат – задает формат исходных данных.

Допустимые значения опции формат приведены в табл. 4.13.

Таблица 4.13 Опции функции Формат

Константа	*Описание*
VbCFText	текст
VbCFBitmap	растровое изображение (*.bmp)
VbCFMetaFile	метафайл(*.Wmf)
VbCFFMetaFile	расширенный метафайл(*.EMF)
VbCFDIB	независимое от устройства растровое изображение(*.dib)
VbCFPalette	цветовая палитра
VbCFRtf	файл в формате*.rtf

Метод ***GetData()*** – восстанавливает данные из объекта ClipBoard.

Синтаксис метода: Объект.GetData [,формат]

Метод ***GetFormat*** – возвращает логическое значение, подтверждающее, хранятся ли в объекте данные указанного формата.

Синтаксис: Объект.GetFormat[,формат]

Метод ***Clear*** – очистка экрана.

Пример 4.41. Анимационное перемещение изображения, содержащегося в элементе управления Image1.Picture, в элемент управления Image2.Picture (рис.4.57):

```
Private Sub Command1_Click()
  Clipboard.Clear
  Clipboard.SetData Image1.Picture, vbCFBitmap
  Image2.Picture = Clipboard.GetData()
End Sub
```

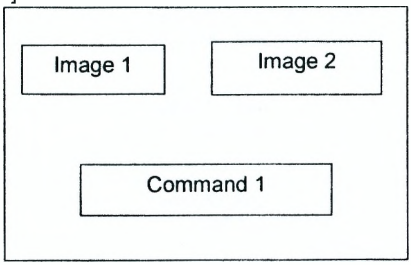


Рис. 4.57. Форма для анимационного перемещения объекта

Анимация посредством присвоения значения свойства одного графического объекта другому

Пример 4.42. "Светофор"

Другой способ анимации состоит в присваивании значения свойства Picture одного

графического элемента управления другому.

На форме (рис. 4.58.) расположен "светофор". На форму помещены семь элементов управления PictureBox или Image. Первый, второй и третий элементы содержат круги зеленого, желтого и красного цвета, соответственно (круги соответствующего цвета можно нарисовать в программе Paint и сохранить в отдельных файлах). Четвертый элемент пустой. Через установленные интервалы времени рисунок из элементов Picture1, Picture2, Picture3, Picture4 перемещается в соответствующие элементы Picture5, Picture6, Picture7 "светофора". Элементы управления Picture1- Picture 4 необходимо сделать невидимыми.

Текст программы

```
Private Sub Form_Click()

Dim i As Integer, Itime As Single
For i = 1 To 10
   Itime = Timer()
    While Timer() - Itime < 1: Wend
  ' пауза
   Picture6.Picture = Picture4.Picture
   Picture7.Picture = Picture4.Picture
   Picture5.Picture = Picture1.Picture
    Itime = Timer()
   While Timer() - Itime < 0.5:  Wend
   'пауза
    Picture5.Picture = Picture4.Picture
    Picture6.Picture = Picture2.Picture
    Itime = Timer()
    While Timer - Itime < 1:
    Wend     'пауза
    Picture6.Picture=Picture4.Picture
    Picture7.Picture = Picture3.Picture
Next i
End Sub
```

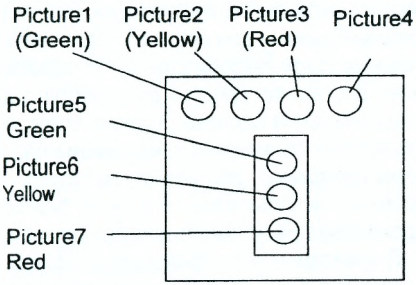


Рис. 4.58. Форма "Светофор"

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Поясните назначение и принцип работы элемента управления Animation.
2. Для чего используется режим DrawMode?
3. Расскажите общий алгоритм создания анимационных эффектов.
4. Перечислите способы создания анимационных эффектов.
5. Как можно организовать паузу?
6. Как создать простую анимацию?
7. Поясните способы создания анимационных эффектов с помощью буфера обмена.
8. Поясните, как создать анимационные эффекты путем обмена значениями между графическими объектами.
9. Поясните принцип работы программы "Мяч" в клетке.
10. Объясните принцип работы программы "Броуновское движение".
11. Объясните принцип работы программы "Светофор".

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с объектом Animation, предназначенным для создания анимационных эффектов и примерами создания пользовательских программ анимации. Для создания пауз исользуются пустые циклы, функции Timer и Time(),

а также объект таймер. Для перемещения объектов могут использоваться метод Move, свойства Top и Left. Гашение изображений при анимации осуществляется путем повторного воспроизведения рисунка в тех же координатах, кода свойству DrawMode присвоено значение 7 (Xor Pen). Для создания анимации могут использоваться также буфер обмена ClipBoard и передача изображений от одного графического объекта другому.

4.10. РАБОТА С ФАЙЛАМИ ДАННЫХ

4.10.1. 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. Visual Basic 6 имеет достаточно средств для работы с данными, но они требуют больших ресурсов вычислительной системы. Однако класс решаемых задач, порой, не оправдывает использования полноценного механизма баз данных, так как это может привести к усложнению программы, увеличению ее размера и замедлению работы. В таком случае целесообразно использовать текстовые файлы. Visual Basic 6.0, так же как и предыдущие версии языка Basic, имеет достаточно средств для работы с текстовыми файлами.

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 (Ввод) 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 новый способ 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. Имя файла формируется по правилам операционной системы Windows.

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 предназначен, обычно, 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.

Так как пользователь при написании программы в принципе не может знать, сколько каналов занято и каков номер свободного канала, то для определения номера свободного канала следует использовать функцию ***FreeFile***. Функция FreeFile возвращает номер свободного канала.

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]. Но этого может не произойти из-за собственных методов буферизации Windows. По этой причине внезапная потеря напряжения в то время, когда файл открыт, почти неизбежно ведет к потере информации и иногда даже повреждает диск.

С целью надежного сохранения информации рекомендуется использовать вместо команды Close команду **Reset**. Эта команда, в отличие от команды Close, дает указание операционной системе сбросить содержимое буфера на диск.

С целью надежного сохранения информации на диске рекомендуется использовать вместо команды Close команду ***Reset.***

Команды записи данных в файл и чтения информации из файлов данных зависят от типа файла.

4.10.2. 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: создание файла и использование 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 операторы **Print # и Write #.** Синтаксис операторов:

Print #<номер канала>, <список переменных>
Write #<номер канала>, <список переменных>

Числовые данные необходимо преобразовывать в строку символов, особенно это касается вещественных чисел, так как десятичную точку программа воспринимает как разделитель данных.

Синтаксис оператора Write # такой же, как и у оператора Print #, но если оператор Print # сохраняет данные в виде обычного текста, то оператор Write # заключает текстовые строки в кавычки, а цифры выводятся без кавычек:

```
Print #1, "Анна", "Минск", 17 , 3.75
```

В файле будет: Анна Минск 17 3 75

```
Write #1, "Анна", "Минск", 17 , 3.75
```

В файле будет: "Анна", "Минск", 17, 3.75

Поэтому 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 #* .

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.

Пример 4.43 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 4.44. Использование файла последовательного доступа

```
Open "R:Test.dan" For Input As #1
    Input A$ ' чтение данных из файла
    Print A$        ' вывод данных на форму
Close #1
```

На форме 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 (C8)
Close #1
```

На форме 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 #.

Создание базы данных с использованием файла последовательного доступа

Базы данных предназначены для хранения структурированных данных. Они представляют собой набор взаимосвязанных таблиц и программы для манипулирования данными (СУБД). На персональных компьютерах чаще всего создаются реляционные базы данных. Реляционные базы данных представляют собой двухмерные таблицы (подробнее см. раздел 6.9). В простейшем случае база данных может включать одну таблицу. Каждая строка такой таблицы представляет собой запись. Первая строка таблицы – строка заголовков. Каждая запись состоит из полей. Каждое поле имеет имя. Поля в таблице образуют колонки. В каждой колонке хранятся данные одного типа.

Под ***структурой базы*** данных понимают состав полей, их имена, типы, размеры в символах.

Пример 4.45. Создание базы данных

В качестве примера создадим базу данных для учета успеваемости студентов. База данных должна содержать следующие поля: Номер по порядку, Фамилия и инициалы, Оценки по предметам обучения (Физика, Математика, Информатика) и Средний балл успеваемости за сессию. База данных должна обеспечивать ввод данных с их визуализацией, сохранение данных на диске, чтение данных с диска и вывод результатов на печать.

Порядок работы.

1. Изобразим структуру базы данных (рис 4.59):

N	Фамилия и инициалы	Оценки			Средний балл
		Физика	Математика	Информатика	

Рис. 4.59. Структура базы данных

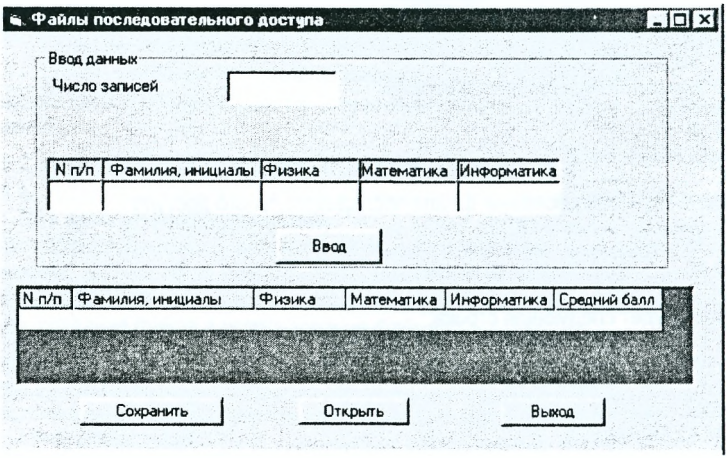


Рис. 4.60. Форма базы переданных "Успеваемость"

Таблица 4.11 Описание переменных

Имя переменной	*Тип переменной*	*Видимость переменной*	*Комментарий*
i, j, k	Integer	Локальная	Используются в качестве переменных цикла
n	Integer	Глобальная	Число студентов в группе
Sb	Single	Глобальная	Средний балл
Bd(5,n)	Single	Глобальная	Массив для хранения данных
nKanal	Integer	Глобальная	Номер канала

Для хранения базы данных в ОЗУ будем использовать двухмерный массив BD(n,5). Где *n* – число записей в базе данных, а 5 – число полей. Номер записи нужен только на экране или на бумаге, в программе хранить его не требуется. Для отображения базы

данных на экране воспользуемся сеткой MSFlexGrid. Для ввода данных создадим линейку из массива элементов управления.

2. Разработаем эскиз формы базы данных "Успеваемость" (рис. 4.60).
3. Опишем переменные, используемые в программе (табл.4.11)
4. Опишем состав элементов управления на форме (табл. 4.12).
5. Напишем текст программы.

Объявление переменных уровня формы:

```
Dim i As Integer, j As Integer, Bd() As String
Dim n As Integer, Sb As Single, nKanal As Integer
```

Установка начальных параметров при загрузке формы:

```
Private Sub Form_Load()
   Me.Height = 5000
   Me.Width = 8000
   For i = 0 To 4
       Grid1.ColAlignment(i)=3
       Grid1.TextMatrix(0,i) = Label2(i).Caption
   Next i
   Grid1. ColAlignment(5) = 3
   Grid1.TextMatrix(0,5) = "Средний балл"
   Grid1.Row = 0
```

Таблица 4.12 Описание элементов управления

Тип	*Имя*	*Назначение*
Label	lblLabel1 lblLabel2()	текст "Число записей" массив элементов управления. Текст – заголовки шапки таблицы ввода данных
TextBox	txtText1 txtText2( )	ввод числа записей массив элементов управления. Поля для ввода данных
MSFlexGrid	Grid1	таблица для вывода результатов
Command	cmdVvod cmdSave cmdOpen cmdExit	ввод данных в массив сохранение данных чтение данных с диска (открытие файла) выход

```
   Grid1.ColWidth(0) = 600
   Grid1.ColWidth(1) = 2000
   Grid1.ColWidth(2) = 1000
   Grid1.ColWidth(3) = 1100
   Grid1.ColWidth(4) = 1200
   Grid1.ColWidth(5) = 1200
End Sub
```

Текст программы для установки числа строк запишем в обработчик события *Change* элемента управления Text1, так как первой операцией при вводе данных необходимо указать число записей.

```
Private Sub txtText1_Change()
   n = Val(txtText1.Text)
   Grid1.Rows = n + 1
End Sub
```

Программы ввода данных, сохранения и чтения данных с диска запишем в обработчики событий соответствующих кнопок.

Процедура ввода данных.

```
Private Sub cmdVvod_Click()
      Dim i As Integer, j As Integer
     'проверка правильности установки числа строк сетки
          If n = 0 Then
```

```
            MsgBox "Укажите число записей"
            Exit Sub
        End If
        If n < Val(txtText2(0).Text) Then
            n = Val(txtText2(0).Text)
            Grid1.Rows = n + 1
        End If
        ReDim Preserve Bd(5, n) As String
        ' ввод данных
        i = Val(txtText2(0).Text)
        For j = 0 To 4
            Bd(j, i) = txtText2(j).Text
            Grid1.TextMatrix(i, j) = Bd(j, i)
        Next j
    Sb = (Val(txtText2(2).Text) + Val(txtText2(3).Text) + Val(txtText2(4).Text)) / 3
        Sb = Round(Sb, 2)
        Grid1.TextMatrix(i, 5) = Str$(Sb)
        Bd(5, i) = Str$(Sb)
        For j = 1 To 4
            txtText2(j).Text = ""
        Next j
        If i < n Then txtText2(0).Text = i + 1
    End Sub
```

Процедура сохранения данных на диске

```
    Private Sub cmdSave_Click()
        nKanal = FreeFile
        Open "r:\Laborat\VisualBasic\file.dan" For Output As #nKanal
        Write #nKanal, n
        For i = 1 To n
          For j = 0 To 5
            Write #nKanal, Round(bd(j, i),2)
          Next j
        Next i
        Close #nKanal
    End Sub
```

Процедура чтения данных (открытие файла)

```
    Private Sub cmdOpen_Click()
      nKanal = FreeFile
      Open "r:\Laborat\VisualBasic\file.dan" For Input As #nKanal
        Input #nKanal, n
        ReDim Bd(5, n)
        Grid1.Rows = n + 1
        For i = 1 To n
          For j = 0 To 5
            Input #nKanal, Bd(j, i)
            Grid1.TextMatrix(i,j) = Bd(j, i)
          Next j
        Next i
        Close #nKanal
    End Sub
```

Процедура завершения работы с программой

```
    Private Sub cmdExit_Click()
      Unload Me
    End Sub
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие типы файлов данных Вам известны и чем они отличаются?
2. Приведите синтаксис команды Open.
3. Приведите синтаксис команды Close.
4. Какие команды используются для записи данных в файл последовательного доступа?
5. Какие команды используются для чтения данных из файла последовательного доступа?
6. Какая последовательность команд необходима для создания файла последовательного доступа?
7. Какая последовательность команд необходима для использования файла последовательного доступа?
8. Расскажите алгоритм разработки базы данных на основе файла последовательного доступа.

ЗАКЛЮЧЕНИЕ

Файлы данных позволяют достаточно простыми средствами создавать и использовать внешние базы данных для хранения исходных данных и результатов вычислений.

Файлы последовательного доступа представляют собой обычные текстовые файлы. Они обладают одним неоспоримым достоинством: простота создания и использования. Файл последовательного доступа может быть создан и отредактирован любым текстовым редактором. При работе с числовыми данными для записи данных в файл целесообразно использовать оператор Write #.

4.10.3. ФАЙЛЫ ПРЯМОГО ДОСТУПА

Создание файлов прямого доступа

В файле прямого доступа данные организованы в виде записей фиксированной длины. Каждая запись состоит из полей. Для каждого поля указывается имя и размер в байтах. Поэтому для символьных полей тип поля должен быть указан с фиксированной длиной, например, Familij As String*20. Длина числовых полей определяется по типу данных.

Файлы прямого доступа обеспечивают доступ к каждой записи файла по его номеру. Если номер записи не используется, то считывание данных производится последовательно, начиная с первой записи. В этом режиме файл прямого доступа превращается в файл с последовательным доступом.

Открытие файла прямого доступа.

```
Open "спецификация_файла" For Random  [Access доступ] [Lock блокировка]
        As[ # ] N  [ Len = длина]
```

Определение длины записи довольно трудоемкая задача. Однако эту задачу можно облегчить, если использовать функцию ***Len (переменная).***

Запись данных в файл прямого доступа осуществляется командой

```
Put # НомерФайла, НомерЗаписи, Переменная.
```

Для чтения данных из файла служит оператор ***Get:***

```
Get # НомерФайла, НомерЗаписи, Переменная.
```

Для чтения и записи данных в файл прямого доступа используется переменная ***пользовательского типа***. Объявление переменной пользовательского типа осуществляется оператором ***Type / End Type***.

Пример 4.46. Объявление пользовательской переменной

Объявить переменную ***Каталог*** пользовательского типа. Переменная должна хра-

нить значения фамилии автора, двух соавторов, наименование книги, издательство, год издания, число страниц.

Порядок работы.

1. Объявим переменную Каталог:

Названия элементов списка записываются без пробелов.

```
Type Каталог
        ФамилияАвтора As String * 20
        Соавтор1   As String * 20
        Соавтор2   As String * 20
        Наименование As String * 100
        Издательство As String * 15
        ГодИздания As Integer
            ЧислоСтраниц  As Integer
End Type
```

2. А теперь объявим переменную ***Библиотека*** типа Каталог:

```
Dim  Библиотека As Каталог
```

3. Присвоим значения переменной.

Обращение к полям переменной осуществляется так же как и к свойствам элементов управления:

```
Библиотека. ФамилияАвтора = "Чингиз Айтматов"
Библиотека. Наименование = "Буранный полустанок"
Библиотека. Издательство = "М.: Нева"
Библиотека. ГодИздания = 1975
Библиотека.ЧислоСтраниц = 256
```

4. Запись данных в файл

```
Private Sub mnuSave_Click ( )
    Put # 1, 1, Библиотека
End Sub
```

5. Чтение данных из файла

```
' объявление пременных
Dim Автор As String, Наименование As String,
Dim Издательство As String, ГодИздания As Integer
Dim ЧислоСтраниц As Integer
---------------------------------------------------------------
Private Sub mnuOpen_Click ( )
Get #1, 1, Библиотека
Автор = Библиотека. ФамилияАвтора
Наименование = Библиотека. Наименование
Издательство = Библиотека. Издательство
ГодИздания  = Библиотека. ГодИздания
ЧислоСтраниц = Библиотека.ЧислоСтраниц
End Sub
```

Команды и функции для работы с файлами

Команды для работы с файлами

Visual Basic 6 имеет значительное число функций для работы с файлами, которые непосредственно взаимодействуют с операционной системой на низком уровне. Эти функции имитируют команды ОС для работы с файлами и накопителями на компьютере:

FileCopy -- копирует файл из одного места в другое:

```
FileCopy <спецификация файла источника>, <спецификация файла  назначения>
```

Kill - удаление одного или нескольких файлов: Kill " *.bak "

MkDir – создание папки: MkDir "C:\PROBA"

Name - переименование, а также перемещение файлов:

Перемещение файла:

```
Name <спецификация файла источника> As <спецификация файла назначения>
    Name "C:\VB\TEST.BAS" As "C: \ ARCHIV\ TEST.BAS"
```

Переименование файла. При переименовании файла имя диска и маршрут не указываются:

```
Name "C:\VB\TEST.BAS" As "TEST.BAK"
```

RmDir – удаление пустой папки: RmDir "C:\PROBA"

ChDir – замена текущего каталога: ChDir <"маршрут">

ChDrive – замена текущего диска: ChDrive <"диск">

Например:

```
ChDrive "D:"      - переход на диск D:
ChDir "D:\Basic"  - переход в папку Basic диска D:
```

Функции для работы с файлами

Функция ***Dir$*** служит для получения списка файлов и поиска файлов на диске. Синтаксис функции: *Dir$(Шаблон [,параметр].*

```
If Dir$("C:\VYFILE.TXT") = "" Then
    MsgBox "Файл не найден"
End If
```

Функция Dir используется также для вывода списка файлов на диске, однако функция возвращает только первое значение. Для получения всего списка ее необходимо поместить в цикл, например:

```
Private Sub Form_Click()
        Dim d As String
    d = " "
    While d <> ""
        d = CurDir$("c:\")
        Print d
    Wend
End Sub
```

Второй необязательный параметр функции Dir$ используется для указания условий отбора файлов, он может принимать следующие значения:

0 – значение, установленное по умолчанию;

1 – поиск файлов, предназначенных только для чтения;

2 – поиск скрытых файлов;

4 – поиск системных файлов;

8 – возвратить метку тома;

16 – поиск каталогов.

В качестве условий отбора можно задавать одновременно несколько параметров. Например следующий оператор задает поиск и вывод на печать системного, скрытого, помеченного только для чтения файла Io.sys:

```
Debug.Print Dir$("C:\IO.SYS", vbHidden + vbSystem + vbReadOnly)
```

Функция ***CurDir$*** – возвращает строку, содержащую текущую папку. Синтаксис функции: *CurDir$(Drive)*

Первый символ параметра Drive возвращает текущий путь для заданного диска. На-

пример, команда Print CurDir$("C") выведет на форму сообщение C:\Program Files\Microsoft Visual Studio\VB98.

Функция ***GetAttr*** – возвращает атрибуты файла.

Синтаксис функции: *GetAttr (маршрут) As String*

Возвращаемые значения:

0 - обычный;	8 - метка тома;
1 - только для чтения;	16 - каталог;
2 - скрытый;	32 - архивный.
4 - системный;	

Функция ***SetAttr*** устанавливает атрибуты файла

SetAttr (маршрут, атрибут)

Функция ***Shell(путь, стиль_окна)*** используется для запуска любого файла в форматах .com, .exe, .bat или .pif:

```
X= Shell "C:\ WINDOWS\ COMMAND \ Format.com A:"
```

Когда VB 6 запускает программу, он создает новое окно и передает ему фокус ввода. Параметр "стиль окна" может принимать 6 значений:

0 - окно скрыто, но имеет фокус;
1 - обычное окно, с фокусом;
2 - окно, представленное значком, с фокусом;
3 - развернутое окно, с фокусом;
4 - обычное окно, без фокуса;
6 - окно, представленное значком, без фокуса.

Функция ***FileDataTime*** – возвращает дату и время создания файла или его последней модификации:

FileDataTime (спецификация файла)

Объект App

Объект App используется для определения местонахождения файлов.

Свойство Path объекта App позволяет вернуть путь текущего каталога, из которого запущено приложение:

```
sAppPaht =  App.Path
```

Символьной переменной присваивается спецификация текущего каталога.

При необходимости из этой переменной можно извлечь имя диска:

```
disk = Left(sAppPath,2)
```

Этим примером из переменной sAppPath извлекаются два симола слева – имя диска, например, D:.

Конструкция sAppPath не возвращает замыкающий символ "\" маршрута, поэтому при необходимости обратиться к файлу в текущем каталоге перед именем файла необходимо добавить обратный слэш. Чтобы избавиться от такого неудобства, этот символ необходимо добавить к переменной sAppPath заблаговременно следующей программной строкой:

```
If Right$(sAppPath,1)<> "\" Then  sAppPath = sAppPath & "\"
```

Этой командой проверяется, если последний символ переменной sAppPath не равен "\", то к указанной переменной прибавляется символ "\".

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем состоит отличие файла прямого доступа от файлов последовательного доступа?
2. Приведите синтаксис команды для открытия файла прямого доступа.
3. Как записать данные в файл прямого доступа?
4. Как прочитать данные из файла прямого доступа?
5. Как объявить переменную пользовательского типа?
6. Назовите команды для работы с файлами?
7. Перечислите функции, используемые для работы с файлами.

ЗАКЛЮЧЕНИЕ

Основным достоинством файлов прямого доступа является возможность вносить записи в файл или считывать записи из файла по номеру записи. Для записи и считывания данных из файла прямого доступа необходимо объявлять переменную пользовательского вида. При большом числе полей в базе данных использование пользовательской переменной становится неудобным.

Visual Basic имеет большое число функций для работы с файлами, которые позволяют реализовать практически все возможности операционной системы по работе с файлами: чтение и поиск папок и каталогов, копирование, пересылка, переименование, удаление файлов, переход на другой диск или каталог.

5. ТЕКСТОВЫЙ ПРОЦЕССОР MICROSOFT WORD 2000

5. 1. НАЧАЛЬНЫЕ СВЕДЕНИЯ

5.1.1. ОКНО ПРОГРАММЫ

Текстовый процессор Microsoft Word 2000 представляет собой интегрированную среду для создания и редактирования документов сложной структуры. Он обеспечивает ввод, редактирование и форматирование текста, вставку диаграмм, таблиц и рисунков, обмен данными с другими приложениями Windows, например Excel, работу с гипертекстовыми документами, просмотр Web-страниц и размещение документов на Web-страницах, подготовку писем и их рассылку по электронной почте. Наличие средств форматирования текста, вставки фотографий и рисунков позволяет использовать Word 2000 как малую издательскую систему. Особенностью текстового процессора, в отличие от текстовых редакторов, является возможность не только вводить, редактировать и выводить текст на печать, но и форматировать текст документа.

Запуск программы осуществляется через команду ***Программы*** главного меню или щелчком мыши по кнопке *W* панели Microsoft Office.

Выход из программы осуществляется командой ***Файл, Выход*** главного меню или комбинацией клавиш *Alt + F4*.

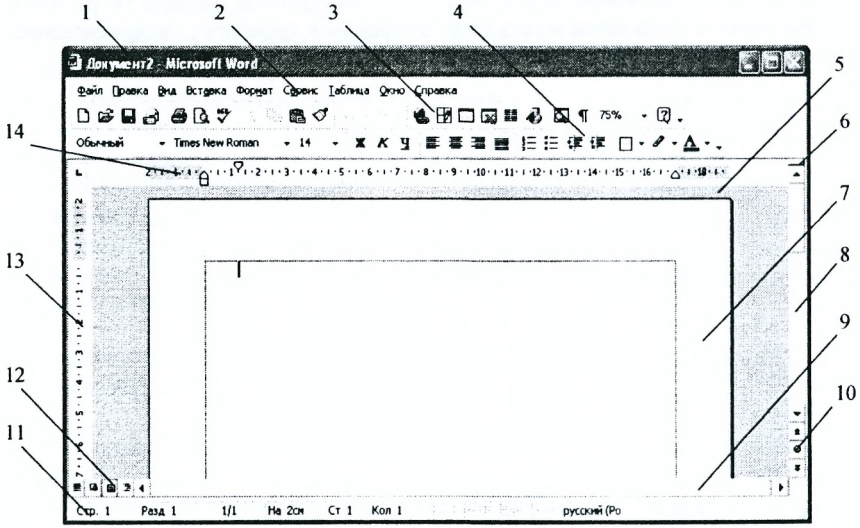


Рис. 5.1. Окно программы Word 2000: 1- строка заголовка, 2 - меню , 3 - стандартная панель инструментов, 4 - панель инструментов форматирования, 5 - окно документа, 6 – кнопка деления окна, 7- документ, 8, 9 - вертикальная и горизонтальная линейки прокрутки, 10 - кнопка перехода к объекту, 11 - строка состояния, 12 - кнопки выбора режима просмотра, 13, 14 - вертикальная и горизонтальная линейки.

Окно программы Word (рис. 5.1) похоже на окна других приложений Windows. В верхней части расположена строка заголовка (1), в которой выводится имя редактируемого документа (по умолчанию предлагается имя *Документа X*) и название программы. В левой части строки заголовка размещается кнопка системного меню, а в правой – кнопки свертывания, развертывания/ восстановления и закрытия окна.

Ниже строки заголовка расположено меню программы (2), стандартная панель инструментов (3), панель инструментов форматирования (4), окно документа (5), документ (7). В окне документа расположены вертикальная и горизонтальная линейки прокрутки (8, 9). Между вертикальной линейкой прокрутки и панелями инструментов расположена кнопка управления делением окна документа разделительной линией (6). На вертикальной линейке прокрутки расположена Кнопка перехода к объекту (10). В нижней части экрана расположена Строка состояния (11). Слева от горизонтальной линейки прокрутки расположены кнопки переключения режима редактирования документа (12). В окно документа могут быть выведены также вертикальная и горизонтальная линейки (13,14).

Меню программы

Чтобы узнать возможности программы, достаточно внимательно изучить меню программы. Главное меню программы горизонтальное. При щелчке мышью по пункту меню открывается меню второго уровня – вертикальное. Если в строке меню имеется стрелка, направленная вправо, то это означает, что данный пункт меню имеет подменю следующего уровня. Если список пунктов меню заканчивается двойной стрелкой, направленной вниз, это означает, что список имеет продолжение. Щелкните по стрелке мышью – список пунктов меню откроется полностью.

Файл – обеспечивает связь с внешними устройствами: открытие файлов, сохранение файлов на диске, настройка параметров страницы, печать страницы, просмотр Web – страниц, отправка сообщений по электронной почте, выход из программы. В данном меню хранится также список последних файлов, с которыми работал пользователь, обычно четыре файла. Число этих файлов можно изменить командой ***Сервис, Параметры***, выбрать закладку ***Общие***, установить флажок ***Помнить список из***... и указать число запоминаемых файлов.

Правка – служит для редактирования файлов: копирования, вставки, удаления, поиска и замены слов и фрагментов текста, восстановления последних удалений текста.

Вид – содержит команды для настройки внешнего вида рабочего окна: установка режимов просмотра документов, вывод на экран и удаление панелей инструментов, оформления колонтитулов, масштаб изображения.

Вставка – позволяет вставлять в документ дату и время, номера страниц, сноски, комментарии, оглавление и указатели, специальные символы, отсутствующие на клавиатуре, рисунки и другие объекты (рис. 5.2).

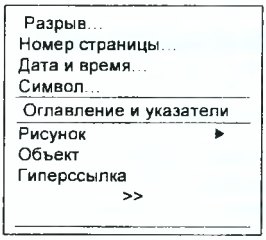


Рис. 5.2. Меню второго уровня пункта главного меню Вставка

Формат – обеспечивает настройку параметров шрифтов, абзацев, колонок, списков, позволяет оформлять текст с помощью рамок и заливок, настраивать стиль документа.

Таблица – позволяет вставлять в текст документа и редактировать таблицы.

Сервис – содержит команды для проверки орфографии, настройки режима переноса слов, подготовки писем, конвертов, наклеек, а также обеспечивает доступ к средствам настройки среды пользователя и автоматизации редактирования текстов с помощью макрокоманд (макросов).

Окно – позволяет разделить рабочее окно по горизонтали на два окна. В обоих окнах помещается текст одного и того же документа. В каждом из этих окон можно просматривать и редактировать текст документа независимо от другого окна. В этом меню имеется также список всех открытых документов, который позволяет быстро переходить от одного документа к другому.

Разделить окно документа можно также с помощью Кнопки деления окна (рис.5.1. (6)). Если зацепить мышью эту кнопку и протащить ее вниз, то появляется линия деле-

ния окна документа на две части по горизонтали. Для отмены деления окна воспользуйтесь командой ***Снять разделение*** меню ***Окно*** или зацепите линию раздела мышью и переместите ее к верхнему краю окна документа.

Панели управления

По умолчанию в окне редактора Word присутствуют две панели инструментов: Стандартная и Форматирование. Стандартная панель инструментов содержит кнопки, дублирующие основные команды главного меню (рис. 5.3).

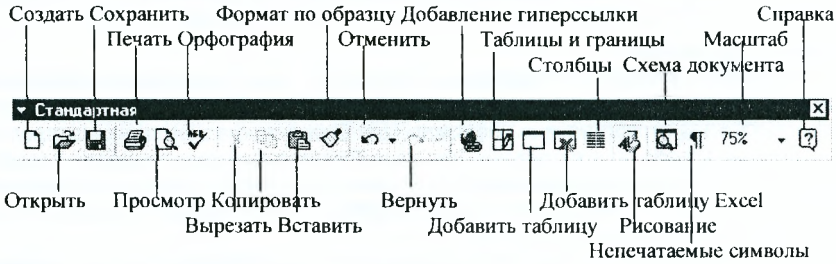


Рис. 5.3. Стандартная панель инструментов

Панель инструментов форматирования содержит кнопки для форматирования текста и абзацев (рис. 5.4). Кнопки x^2, x_2, не обозначенные на рис. 5.4, служат для создания надстрочного и подстрочного текста, соответственно. Кнопка Р/А – нестандартная, предназначена для исправления текста: русские символы преобразует в латинские и наоборот, латинские символы преобразует в русские в соответствии с раскладкой клавиатуры (см. раздел 5.9).

Число видимых панелей инструментов может устанавливаться пользователем. Для этого необходимо ввести команду ***Вид, Панели инструментов*** и

Рис. 5.4. Панель инструментов Форматирование

установить флажок у панели, которую необходимо отобразить на экране, или снять флажок у панели, которую нужно удалить с экрана. То же можно сделать с помощью контекстного меню. Щелкните правой кнопкой мыши по одной из панелей инструментов – откроется контекстное меню со списком панелей инструментов.

Режимы редактирования документов

Имеется четыре режима редактирования: Обычный, Web-документа, Разметки страниц и Структуры документа.

Обычный режим используется для ввода и редактирования текста. Стили, колонтитулы, рисунки и колонки в этом режиме не отображаются. Однако в этом режиме отображаются линии разрыва страниц и разделов. Линии разрыва разделов можно удалить: выдерите линию раздела мышью и нажмите клавишу Delete.

Режим Web-документа позволяет просматривать Web-страницы. В этом режиме кнопки режимов просмотра документа не отображаются.

Режим разметки страниц позволяет отобразить все элементы форматирования и все объекты, внедряемые в документ. В этом режиме все элементы отображаются так, как будут выглядеть при печати. Поэтому этот режим можно считать основным режимом для ввода и редактирования текста.

Режим структуры позволяет создать структуру документа. В этом режиме можно просматривать структуру документа, менять уровень просмотра, перемещать целые разделы путем перетаскивания заголовков, изменять уровень заголовков. Если активна кнопка "Все заголовки", то отображается весь текст документа.

Управлять выбором режима просмотра документов можно с помощью команд меню Вид или кнопок выбора режима просмотра.

Линейки

Горизонтальные и вертикальные линейки служат для позиционирования табуляторов, с помощью которых можно управлять размещением текста на странице, а также позволяют настраивать некоторые параметры страниц и абзацев.

Строка состояния

Строка состояния служит для отображения состояния редактируемого документа. В ней отображается номер текущей страницы и раздела, номер текущей страницы и общее число страниц в документе, позиция точки вставки (расстояние от верхнего края документа, номер строки и номер колонки), а также кнопки ЗАП, ИСПР, ВДЛ, ЗАМ. Кнопка ЗАП предназначена для записи пользовательских программ автоматизации редактирования документа (макросов). Кнопка ИСПР служит для включения режима автоматического исправления текста и опечаток. Кнопка ВДЛ – включает режим выделения текста с помощью клавиш управления перемещением курсора без нажатия клавиши Shift. Кнопка ЗАМ – переключает режим вставки и замещения текста. Следующие две кнопки служат для выбора словаря проверки правописания и состояния проверки правописания. Управление этими кнопками осуществляется двойным щелчком мыши.

Линейки прокрутки

Линейки прокрутки служат для просмотра документа. На вертикальной линейке прокрутки имеется несколько кнопок. Кнопки с одиночной стрелкой служат для перемещения текста на одну строку в соответствующем направлении, двойные стрелки перемещают текст на страницу. Быстрое перемещение по тексту удобно с помощью ползунка: зацепите ползунок мышью и перемещайте в требуемом направлении. При этом слева от ползунка появляется всплывающее окно сообщения, в котором отображается номер страницы и наименование раздела.

На вертикальной линейке прокрутки в Word 2000 появилась новая кнопка – Выбор объекта перехода. При щелчке мышью по этой кнопке открывается меню, в котором необходимо выбрать объект для быстрого перехода.

5.1.2. СОХРАНЕНИЕ И ОТКРЫТИЕ ДОКУМЕНТОВ

Первое, о чем должен позаботиться пользователь, работая в редакторе Word, это о сохранении набранного текста. В случае аварийного выключения питания или каких-либо неисправностей, имеется опасность потерять результаты всей работы. Поэтому сразу же после загрузки редактора и открытия нового документа сохраните его на диске командой ***Файл***, ***Сохранить*** или ***Файл, Сохранить как***. При этом открывается окно диалога (рис. 5.5).

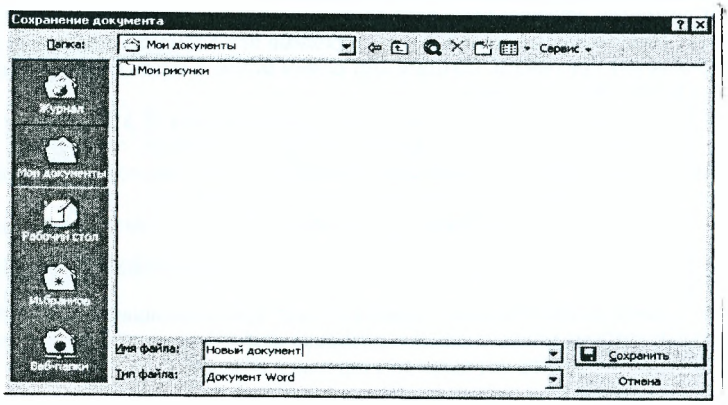


Рис. 5.5. Сохранение документа

Окно сохранения документов стандартное для всех приложений Windows. В верхней части окна диалога расположен список "Папка". Откройте список, выберите нужный диск и папку для сохранения файла. Справа от раскрывающегося списка "Папка" расположено несколько кнопок: "Назад" – для перемещения назад по пройденному ранее маршруту, "Перейти на один уровень вверх" – для выхода из текущей папки, "Найти в Web" – для поиска в Internet, "Удалить" – для удаления выделенной папки или файла, "Создать папку" – для создания новой папки, "Представления" – для управления представлением файлов в окне диалога, сортировки файлов и др., "Сервис" – дополнительные возможности по работе с файлами и настройки режима сохранения файлов.

В нижней части окна диалога расположены два списка: Имя файла и Тип файла. В строке ввода Имя файла набирается имя файла или выбирается из списка. Тип файла выбирается из списка. По умолчанию предлагается тип файла "Документ Word", расширение имени файла .doc.

В процессе работы целесообразно периодически сохранять информацию на диске, чтобы минимизировать возможные потери информации. Для этого можно установить режим автосохранения: введите команду ***Параметры*** из меню ***Сервис***, откройте закладку ***Сохранение*** и установите флажок ***Автосохранение каждые***, а также установите интервал времени, через который будет выполняться автосохранение документа. Желательно также установить флажок *Автосохранение в фоновом режиме*, чтобы программа не запрашивала вашего согласия на сохранение документа.

Открытие существующих документов осуществляется с помощью команды ***Файл, Открыть.*** Окно диалога для открытия файла аналогично окну диалога сохранения файла (рис. 5.5).

5.1.3. ВВОД И РЕДАКТИРОВАНИЕ ТЕКСТА

Ввод текста

Текст вводится после точки ввода, которая отображается мигающей вертикальной чертой (курсор). Конец абзаца или текста обозначается специальным символом ¶. Чтобы показать символы конца абзацев следует активизировать кнопку "Непечатаемые символы" на панели инструментов стандартной. Текст можно вводить в любой точке ра-

бочего окна, для этого щелкните в нужном месте рабочего окна дважды мышью.

При вводе длинного текста он автоматически переносится на другую строку. При нажатии клавиши Enter курсор ввода переходит на другую строку в положение абзацного отступа. Для вставки пустой строки нажмите клавишу Enter.

Чтобы разбить строку на две строки, установите курсор в точку раздела и нажмите клавишу Enter. Для объединения строк установите курсор в конец первой строки и нажмите клавишу Delete (Del) или установите курсор в начало второй строки и нажмите клавишу BackSpace (Возврат на шаг).

Выделение текста

Все операции в редакторе: копирование, перемещение, удаление, - выполняются над выделенным текстом. Для выделения текста можно использовать несколько способов:

- отдельное слово выделяется двойным щелчком мыши;
- абзац выделяется тройным щелчком мыши;
- другой способ выделения абзаца – установите указатель мыши на поле выделения и протяните мышью по этому полю до последней строки выделяемого текста;
- для выделения произвольного фрагмента текста установите указатель мыши в начало выделяемого текста и протяните мышью до конца выделяемого текста. То же можно сделать с помощью клавиш управления перемещением курсора при нажатой клавише Shift.

Перемещение или копирование текста

Для копирования текста можно пользоваться командами ***Копирование*** и ***Вставка*** ме-

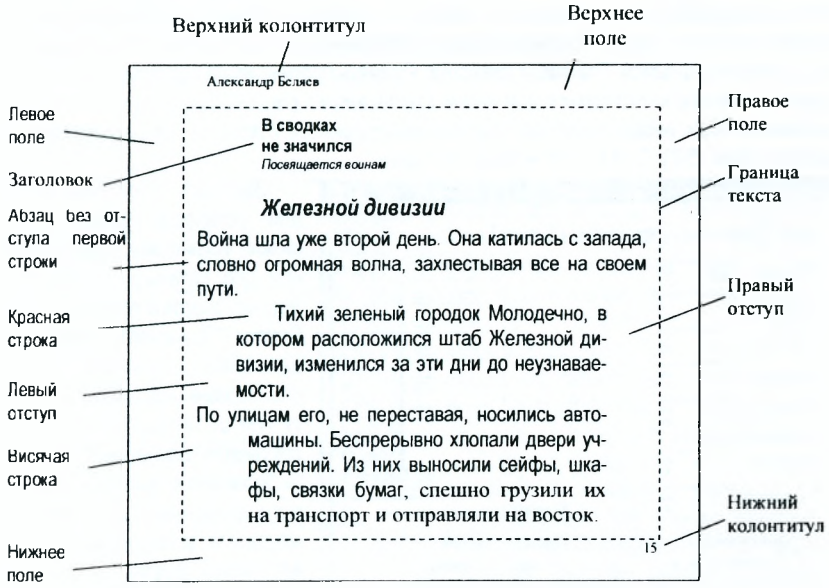


Рис. 5.6. Фрагмент оформления страницы документа

ню Правка, кнопками Копирование и Вставка панели инструментов Стандартная или использовать прием перетаскивания выделенного текста мышью при нажатой клавише Ctrl.

При копировании текста, таблиц, рисунков и других объектов с помощью команд меню или панели инструментов следует придерживаться следующего алгоритма:

- выделите копируемый текст или объект;
- введите команду ***Копировать***. Выделенный текст помещается в буфер обмена. Из буфера обмена текст может быть вставлен в любое место текущего документа или другого документа;
- переместите курсор в место вставки;
- введите команду ***Вставить***.

Перемещение отличается от копирования тем, что исходный текст удаляется. В этом случае для помещения текста (объекта) в буфер целесообразно использовать команду ***Вырезать*** меню ***Правка*** или кнопку Вырезать панели инструментов. При перемещении объекта мышью клавиша Ctrl не нажимается.

Поля

Документ, подготовленный пользователем, содержит ряд элементов: поля, заголовки, форматированный текст, абзацы, колонтитулы, таблицы, рисунки, графики, формулы и т.п.. Основные элементы документа приведены на рис. 5.6.

Поля ограничивают текст документа. Левое поле используется обычно для подшивки и принимается равным 20 – 25 мм, правое поле – не менее 10 мм. Верхнее и нижнее поля используются для размещения колонтитулов. Ширина верхнего поля принимается равной 20 мм, а нижнего поля – 20 – 25 мм. В редакторе Word предусмотрена возможность, для удобства пользователя, показать визуально границы текста. Для этого необходимо выбрать в меню ***Сервис*** команду ***Параметры*** и на закладке ***"Вид"*** окна диалога установить флажок ***Границы текста***. Размеры полей можно регулировать. Для этой цели можно воспользоваться командой ***Параметры страницы*** из меню ***Файл*** или линейками. При вводе команды Параметры страницы открывается одноименное окно диалога (рис. 5.7). Оно имеет четыре закладки.

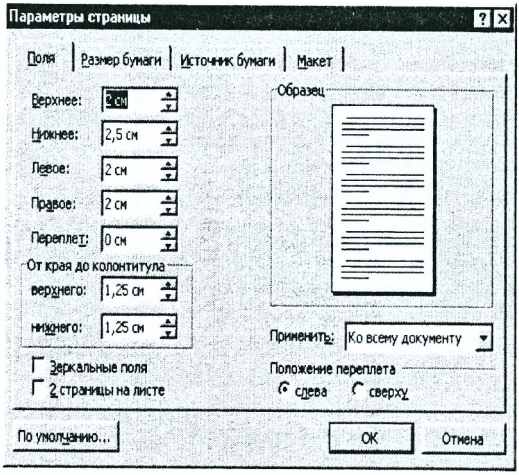


Рис. 5.7. Окно диалога Параметры страницы

Закладка "Поля" предназначена для установки размеров полей. Можно установить зеркальные поля или две страницы на листе с помощью соответствующих флажков. Список "Применить" позволяет задать область действия установленных параметров: ко всему документу или от текущего листа до конца документа. Закладка "Размер бумаги" позволяет выбрать размер бумаги из заданных стандартных форматов или установить свой пользовательский формат, а также позволяет установить ориентацию страницы при печати: книжную или альбомную. Закладка "Источник бумаги" дает возможность выбрать

способ подачи бумаги: отдельные листы или с рулона. Закладка "Макет" (рис. 5.6) предназначена для настройки параметров колонтитулов, выравнивания текста на странице, установки границ и нумерации строк.

Заголовки

Заголовки оформляются с помощью стилей заголовков из списка "Стиль", расположенного в левой части панели инструментов форматирования (рис.5.8).

Стили, используемые в заголовках, не должны использоваться в тексте документа для выделения других фрагментов текста. При оформлении заголовков с помощью стилей открывается возможность автоматизировать процедуру составления оглавления (содержания) документа.

Рис. 5.8. Стили заголовков

Оформление шрифтов

Чтобы сделать текст более читабельным, удобным для чтения и восприятия пользователем, применяют различное оформление шрифта. К параметрам шрифта, подлежащим настройке, относятся: тип шрифта, размер, начертание (полужирный, курсив, подчеркнутый), цвет, межсимвольный интервал и ряд дополнительных эффектов.

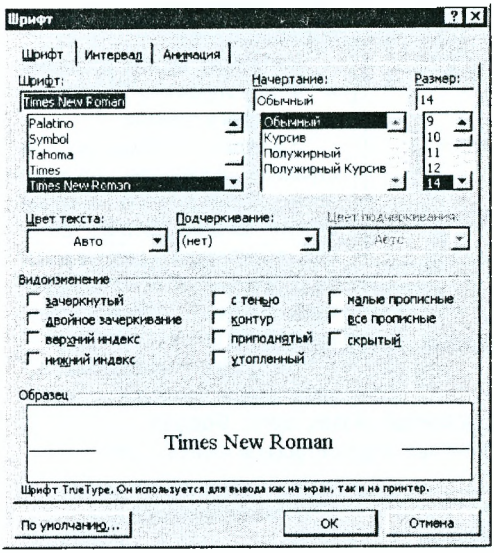


Рис. 5.9. Окно диалога для настройки параметров шрифтов

Настройку параметров шрифтов осуществляют с помощью панели инструментов форматирования или окна диалога Шрифт (рис.5.9), которое вызывается на экран командой ***Шрифт*** меню ***Формат***. Окно диалога имеет три закладки: "Шрифт", "Интервал" и "Анимация". На закладке "Шрифт" настраиваются практически все параметры шрифтов. Все изменения немедленно отображаются в окне "Образец". Для печатного текста наиболее подходящими являются шрифты Times New Roman – шрифт с засечками и Arial – прямоугольный шрифт. Не все шрифты имеют в своем составе русские символы. На закладке Интервал нас интересует один параметр – межсимвольный интервал. Он может быть обычным, разреженным или уплотненным. Для разреженного и уплотненного шрифта предусмотрена возможность регулировки степени разрядки или уплотнения. При этом используется типографская единица измерения – пункт. Анимация (Красные муравьи, Мерцание и т.д.) – отображаются только на экране монитора.

Оформление абзацев

Основной текст состоит из абзацев (рис. 5.6). Абзац – это фрагмент текста, который содержит, как правило, законченную мысль. Для редактора текста абзац – это текст, заключенный между двумя нажатиями клавиши Enter. При нажатии клавиши Enter автоматически вставляется символ возврата каретки. Точка вставки (курсор) переводится на новую строку в положение отступа первой строки. При оформлении абзацев могут настраиваться следующие параметры: уровень текста, выравнивание, отступы от границ текста, первая строка, межстрочный интервал и расстояние между абзацами, положение на странице.

Уровень текста выбирается из списка, он определяет высоту шрифта.

Выравнивание может принимать четыре значения: по левому краю, по центру, по правому краю и по ширине. Выравнивание по ширине предпочтительнее при оформлении документов, так как в этом случае оба края текста будут ровными.

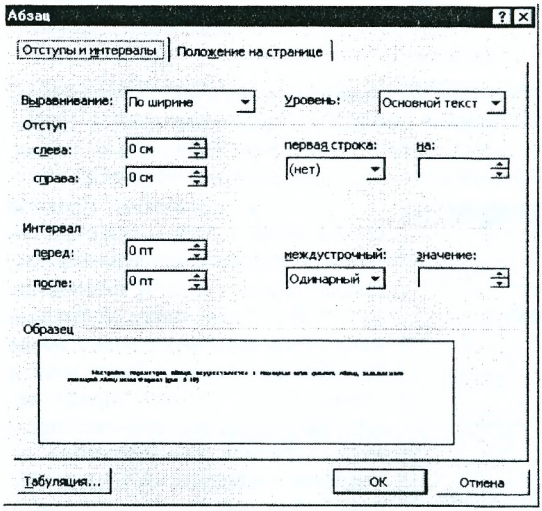


Рис. 5.10. Окно диалога для настройки параметров абзаца

Отступы слева и справа определяют расстояние текста от границы текста, но не от края страницы.

Первая строка может иметь три значения: без выступа, выступ вправо – красная строка или выступ влево – висячая строка (рис. 5.6).

Интервал между абзацами и *междустрочный интервал* измеряются в пунктах.

Закладка "Положение на странице" определяет возможность использования висячей строки, взаимодействие абзацев, возможность разрыва абзаца при переносе на следующую страницу, перенос слов, нумерацию строк.

Настройка параметров абзаца осуществляется с помощью окна диалога Абзац, вызываемого командой ***Абзац*** меню ***Формат*** (рис. 5.10). Часть параметров можно регулировать с помощью горизонтальной линейки (рис. 5.11).

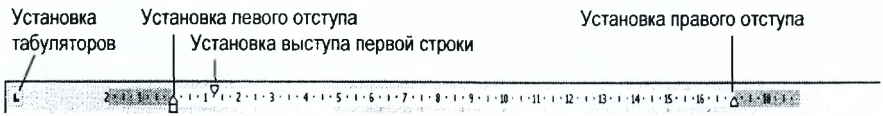


Рис.5.11. Горизонтальная линейка

На горизонтальной линейке имеется три маркера: два в левой части и один в правой части линейки. Эти маркеры позволяют устанавливать отступы и выступ первой строки. Если маркеры "Установка левого отступа" и "Установка выступа первой строки" совмещены, то выступ у первой строки отсутствует, если маркер "Установка выступа

первой строки" находится справа от маркера "Установка левого отступа", то формируется красная строка, а если маркер "Установка выступа первой строки" находится слева от маркера "Установка левого отступа", то формируется висячая строка (рис. 5.12).

На горизонтальной линейке в левой и правой части имеются серые полоски - это установленные значения полей. Размеры полей можно изменить с помощью мыши. Установите указатель мыши на границу раздела поля таким образом, чтобы он принял

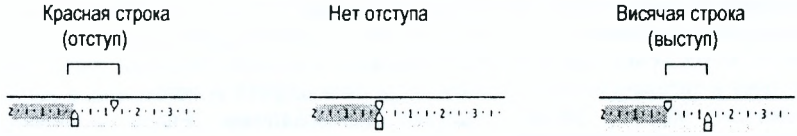


Рис. 5.12. Форматирование абзацев с помощью горизонтальной линейки

вид двухнаправленной стрелки, и перетащите границу раздела в нужном направлении.

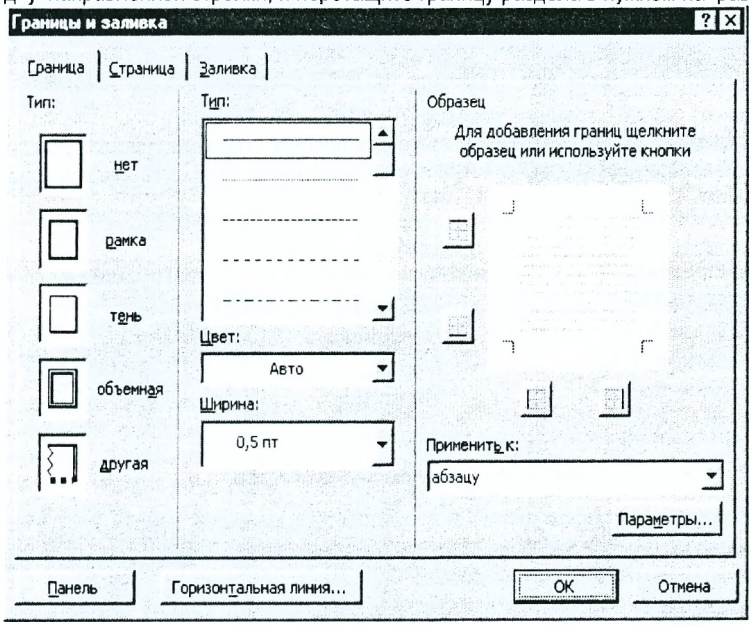


Рис. 5.13. Окно диалога Границы и заливка

Табуляторы

Слева от горизонтальной линейки (рис. 5.11) находится кнопка для установки табуляторов. Табуляторы позволяют управлять размещением текста. По умолчанию текст выравнивается по левому краю, этому состоянию соответствует маркер табуляции - ∟, маркер ⊥ - означает выравнивание по центру , а маркер ⌋ - выравнивание по правому краю, четвертый тип маркера – выравнивание по десятичному разделителю. Маркер применяется к выделенному абзацу или абзацам. Установить позиции табуляции можно также с помощью команды ***Табуляция*** меню ***Формат***.

Границы и заливка

Границы и заливка применяются для выделения текста, оформления страниц и абзацев. Для этой цели можно воспользоваться кнопкой панели инструментов ***Границы***, или командой ***Границы и заливка*** меню ***Формат.***

.Выделите текст, который требуется заключить в рамку и введите команду ***Границы и заливка*** из меню ***Формат*** – открывается одноименное окно диалога (рис. 5.13). Окно диалога имеет три закладки. Закладка *Граница* позволяет установить границы на выделенный фрагмент текста, а закладка *Страница* позволяет установить границы страницы для всего документа или раздела, первой страницы или всех страниц кроме первой. При установке гарниц имеется возможность выбрать тип рамки, тип линии, а также цвет и толщину линии. В правой части окна диалога имеется окно, в котором отображается образец рамки, созданной пользователем. Рядом с образцом расположены четыре кнопки. С помощью этих кнопок можно управлять отображением границ: установить границы с одной, двух, трех или со всех сторон. Закладка *Заливка* позволяет установить цвет фона для абзаца или выделенного фрагмента текста. Для отмены заливки или границ выделите абзац или фрагмент текста, для которого необходимо выполнить удаление границ или заливки, войдите в окно диалога и выберите объект со словом "нет": "Граница, тип" – нет или "Заливка, нет заливки" .

Списки

Редактор Word позволяет автоматизировать процесс создания списков. Списки могут быть нумерованные, маркированные и многоуровневые. Примеры списков приведены на рис. 5.14.

Список		
нумерованный	маркированный	многоуровневый
1. Петров П. П. 2. Иванов Г. К. 3. Цветков П. С. 4. Полежаев Н. Р.	• Клубника • Черешня • Слива • Виноград	1. Глава 1 1.1. Раздел 1 1.1.1. Подраздел 1 1.1.2. Подраздел 2

Рис.5.14. Примеры видов списков

Для оформления списков можно воспользоваться кнопками панели инструментов форматирования Нумерация (или Маркеры) или командой ***Список*** меню ***Формат***. Списки можно офомлять двумя способами:

1. Написать текст, выделить его и ввести команду Формат, Список, выбрать нужный вид списка и щелкнуть по кнопке ОК.

2. Ввести команду Списки из меню Формат, выбрать нужный вид списка, щелкнуть по кнопке ОК. В точке вставки появится первый номер списка. Ввести текст первого элемента списка. Нажать клавишу Enter. При переходе на новую строку автоматически появляется следующий номер.

Окно диалога Список имеет кнопку Изменить. При выборе этой команды откроется окно настройки параметров списков (рис. 5.15), в котором можно выбрать шрифт, формат номера, порядок нумерации, вид маркера, установить отступы для номера и для текста.

5.1.4. ПЕЧАТЬ ДОКУМЕНТОВ

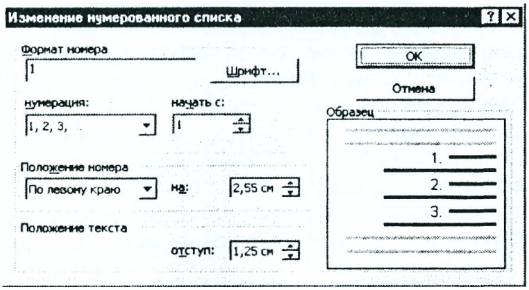


Рис. 5.15. Окно настройки параметров списка

Подготовленный документ можно просмотреть и затем распечатать. Просмотр документа осуществляется командой ***Предварительный просмотр*** меню ***Файл*** или соответствующей кнопкой панели инструментов. В этом режиме документ отображается в том виде, как будет выглядеть на бумаге. При выборе режима Предварительный просмотр автоматически открывается одноименная панель инструментов (рис. 5.16), которая позволяет управлять режимом просмотра: увеличивать или уменьшать масштаб изображения, просматривать одновременно несколько страниц, выводить на экран или убирать с экрана линейки, развертывать лист на весь экран. Представляет интерес кнопка ***Подгонка страниц***. Часто случается, что на последней странице документа размещается всего несколько строк. Кнопка Подгонка страниц позволяет избавиться от таких страниц путем автоматического подбора высоты шрифта. На панели инструментов имеется также кнопка контекстно-зависимой справки, которая позволяет получить подсказку по назначению кнопок на панели инструментов. Активизируйте кнопку Контекстная справка – возле указателя мыши появляется вопросительный знак. Щелкните мышью по интересующей Вас кнопке и на экране появится справ-ка о назначении этой кнопки.

Рис.5.16. Панель инструментов Предварительный просмотр

Для вывода документа на печать необходимо воспользоваться командой ***Печать*** меню ***Файл*** или соответствующей кнопкой панели инструментов. При использовании для печати команды меню Файл на экран выводится диа-логовое окно настройки параметров печати (рис. 5.17). В этом окне можно выбрать, что печатать: весь документ, текущую страницу, указанные номера страниц или выделенный фрагмент текста. Дополнительные возможности управления выводом на печать предоставляют раскрывающиеся списки

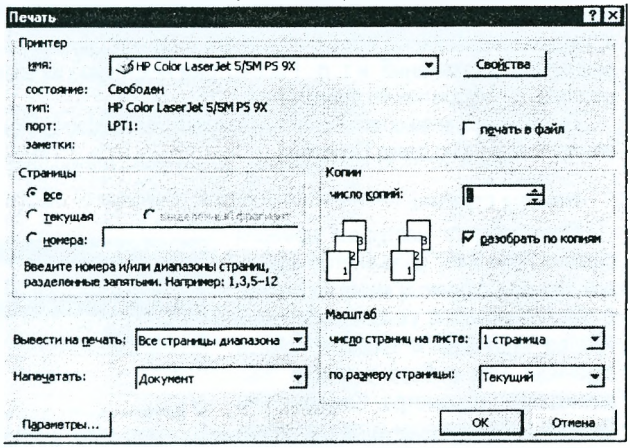


Рис. 5.17. Окно диалога настройки параметров печати

"Вывести на печать" и "Напечатать". Список Вывести на печать позволяет вывести на печать все страницы диапазона, только четные или только нечетные страницы диапазона. Список "Напечатать" позволяет напечатать Документ, Сведения, Примечания, Стили, Элементы автотекста или Сочетания клавиш. Можно указать число копий для печати и порядок печати листов. Если установить флажок "Разобрать по копиям", то будет напечатан сначала первый экземпляр документа, а затем следующий. Если флажок "Разобрать по копиям" снять, то сначала будут напечатаны все экземпляры первой страницы, затем второй страницы и т. д.. В группе "Масштаб" можно указать число страниц на листе, а также указать формат бумаги, на который будет печататься документ. Имеется также возможность выбрать тип принтера и настроить его свойства. При активизации флажка "Печать в файл". Документ будет сохранен в файле на диске.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как установить параметры страницы?
2. Назовите основные приемы выделения текста?
3. Расскажите назначение всех кнопок на панели инструментов Стандартная и Форматирование.
4. Какие параметры шрифтов Вы знаете, как их установить?
5. Какие параметры абзацев Вы знаете, как их установить?
6. Как создать маркированный (нумерованный, иерархический список)?
7. Как изменить параметры списка?
8. Как установить границы абзаца, страницы?
9. Как сохранить документ на диске?
10. Как вывести документ на печать?

5.2. ОФОРМЛЕНИЕ ДОКУМЕНТА

Деловой документ должен быть определенным образом оформлен. В него могут вставляться различные объекты: номера страниц, сноски, рисунки, формулы, таблицы, диаграммы, оглавление и т. п. Рассмотрим некоторые из возможностей текстового процессора по оформлению документов.

Вставка различных объектов в документ, за исключением колонтитулов, осуществляется с помощью меню ***Вставка***.

Номера страниц

Окно диалога настройки номеров страниц (рис.5.18) вызывается командой ***Номера страниц*** меню ***Вставка***. В этом окне можно указать положение номера страницы и формат номера страницы. При выборе в списке "Выравнивание" параметра *Снаружи* или *Внутри* автоматически устанавливаются зеркальные поля. Флажок "Номер на первой странице" рекомендуется снять, так как на первой странице документа номер страницы обычно не ставят.

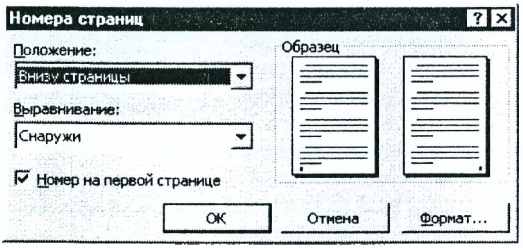


Рис. 5.18. Настройка параметров номеров страниц

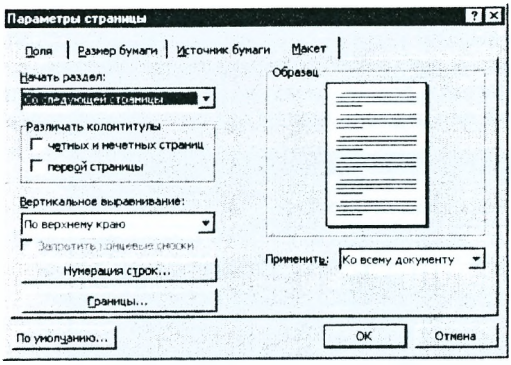


Рис. 5.19. Настройка параметров колонтитулов

Колонтитулы

Колонтитулы – это короткий текст, расположенный в верхнем или нижнем поле страницы. Это может быть краткое сообщение, например, фамилия автора документа, наименование раздела, номер страницы, дата и время разработки документа и т. п. По умолчанию все колонтитулы одинаковые. Однако можно изменить эту настройку. Если установить флажки "Различать колонтитулы четных и нечетных страниц" и "Различать колонтитулы первой страницы", то появится три вида колонтитулов (рис.5.19). На первой странице колонтитулы обычно не пишутся.

Для установки колонтитулов необходимо ввести команду ***Колонтитулы*** из меню ***Вид***. При этом открывается второй слой документа, а основной текст становится недоступным для редактирования. Одновременно на экран выводится панель инструментов

Рис.5.20. Панель инструментов Колонтитулы

для управления колонтитулами (рис. 5.20). При выводе на печать и основной текст и колонтитулы отображаются одинаково.

Название значков на панели инструментов Колонтитулы говорят сами за себя. Кнопка "Параметры страницы" выводит на экран знакомое нам уже окно диалога для управления колонтитулами (рис. 5.19).

Дата и время

Окно диалога "Дата и Время" позволяет вставлять в документ поле даты и времени. Имеется возможность выбрать формат представления даты и времени. Флажок "Обновлять автоматически" позволяет установить режим автоматического обновления даты. Если флажок установлен, то при каждой загрузке документа будет устанавливаться текущая дата.

Вставка специальных символов

Команда ***Символ*** меню ***Вставка*** выводит на экран одноименное окно диалога (рис. 5.21). Оно позволяет выбрать и вставить в текст любой символ из набора символов, установленных на компьютере. Для облегчения поиска символы сгруппированы по типу шрифтов и по назначению. Для этого в окне диалога имеется два окна: "Шрифт" и "На-

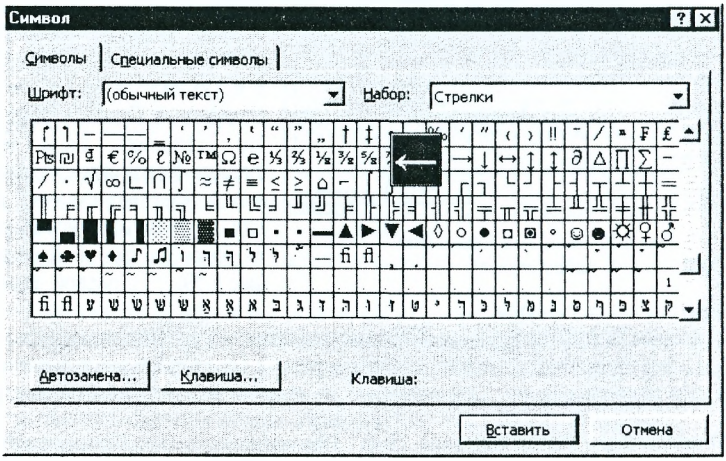


Рис.5.21. Вставка специальных символов

бор". Для вставки нужного символа выделите его курсором (символ увеличивается в размерах и выделяется на синем фоне) и щелкните по кнопке ***Вставить***.

Вставка названий

Команда ***Название*** меню ***Вставка*** позволяет автоматизировать нумерацию Рисунков, Таблиц, Формул. При вводе команды открывается окно диалога "Название" (рис. 5.22). В строке ввода после номера рисунка следует ввести название этого рисунка. Номера рисунков программа формирует автоматически. В группе "Параметры" расположены два списка: "постоянная часть" и "Положение". Список "постоянная часть" содержит список постоянной части названия. Эта часть может изменяться пользователем. Для этого щелкните по кнопке Создать и в открывшемся окне диалога введите название, например, Рис. Это название попадает в список и может использоваться в качестве постоянной части номера рисунка. Список "Положение" позволяет указать место размещения надписи: над выделенным объектом или ниже выделенного объекта.

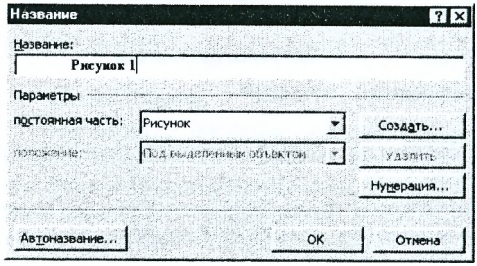


Рис. 5.22. Вставка названий

Сноски

Сноска – это краткий комментарий к тексту, пояснение какого-либо термина, описание событий, связанных с какой-нибудь датой и тому подобное. Различают обычную и концевую сноски. Обычная сноска помещается в конце текущей страницы, концевая сноска – в конце документа. Для вставки сноски установите курсор после текста, к которому будет относиться сноска, введите команду ***Сноска*** из меню ***Вставка***, выберите тип сноски, способ нумерации и щелкните по кнопке ОК. В месте размещения курсора появится номер сноски. Редактируется сноска как обычный текст. Для удаления сноски выделите и удалите ее номер.

Перекрестная ссылка

Перекрестная ссылка связывает текст в документе или разрешает пользователям переходить к нужному тексту или объекту. Имеется два вида ссылок: первый тип ссылки связывает два объекта в одном документе. Второй тип ссылок позволяет переходить от одного документа к другому – это *гиперссылки.*

Первый тип ссылки дает пользователю только информацию, например, "Подробнее об этом можно узнать в разделе Сноски на стр 17". При изменении местоположения раздела пользователь все равно найдет его по названию.

Второй тип ссылки применяется в электронных документах для автоматического перехода к требуемому разделу. Если на поле ссылки навести указатель мыши, то он превращается в руку, а при щелчке мышью по этому полю пользователь перейдет к требуемому разделу. При этом открывается панель инструментов Web. Чтобы вернуться назад, надо щелкнуть мышью по кнопке Назад на панели инструментов Web.

При создании ссылки в документ вставляется поле с выбранным типом ссылки, выделенное серым цветом. Перекрестная ссылка может ссылаться на нумерованные абзацы, заголовки, закладки, сноски, рисунки, таблицы и другие объекты, созданные средствами Word. Например, нельзя сделать сноску на заголовок, если он не оформлен с помощью стиля заголовков, нельзя сослаться на слово Рисунок, если он не создан с помощью команды Название меню Вставка.

Для создания ссылки напишите нужный текст, например, "Подробнее об этом можно узнать в разделе Сноски на стр. 17". Выделите слово Сноски (на этом месте может быть любой текст, так как он будет заменен на выбранный) и введите команду Вставка, Перекрестная ссылка – открывается окно диалога Перекрестная ссылка (рис. 5.23).

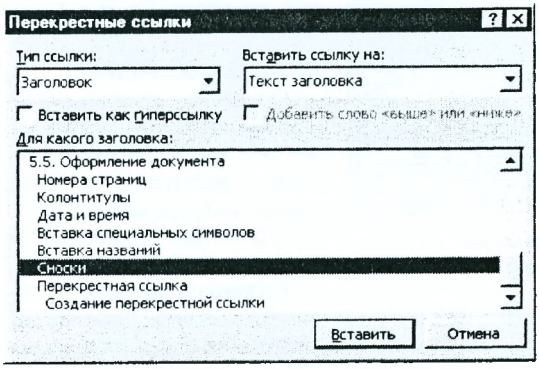


Рис. 5.23 Создание перекрестной ссылки

Выберите в списке "Тип ссылки" требуемый тип, например, Заголовок. В списке "Вставить ссылку на:" выберите объект, на который будите ссылаться: Текст заголовка, номер страницы и др.. А в окне "Для какого заголовка:" выделите заголовок Сноски. Если предполагается создавать гиперссылку, то установите соответствующий флажок в окне диалога. Для завершения работы по созданию перекрестной ссылки щелкните по кнопке Вставить.

Предметный указатель

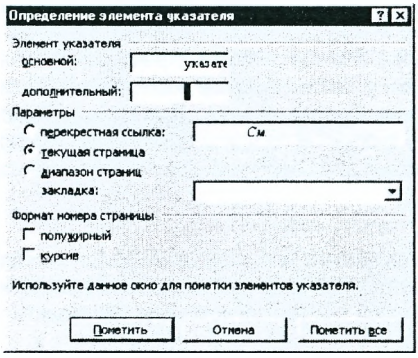


Рис. 5.24. Определение элементов указателя

В больших документах для удобства поиска требуемой информации иногда вставляют указатели – алфавитные списки ключевых слов. Для вставки ключевого слова или фразы в указатели выделите этот текст и нажмите комбинацию клавиш Alt+Shift+X – открывается окно диалога Определение элемента указателя (рис. 5.24). В строке ввода Элемент указателя основной будет записано выделенное слово. К основному элементу указателя можно добавить еще два дополнительных элемента. Введите в строке ввода "Дополнительный" первый дополнительный элемент, а затем через двоеточие – второй дополнительный элемент. Чтобы запомнить выделенные элементы, щелкните по кнопке ***Пометить*** или ***Пометить все***. Во втором случае будут помечены все вхождения ключевого слова в документ.

При необходимости вставить перекрестную ссылку активизируйте соответствующий переключатель и добавьте текст после См... в строке ввода. Для включения в указатель диапазона страниц, выберите сначала этот диапазон и отметьте его закладкой. Затем установите переключатель Диапазон страниц и выберите в списке требуемую закладку.

Для завершения операции по формированию Указателя перейдите на последнюю страницу документа. Вставьте разрыв страниц, введите заголовок для предметного указателя и введите команду ***Вставка***, ***Оглавление и указатели***. Откроется одноименное окно диалога (рис. 5.25). Выберите стиль оформления Предметного указателя и щелкните по кнопке ОК для завершения работы.

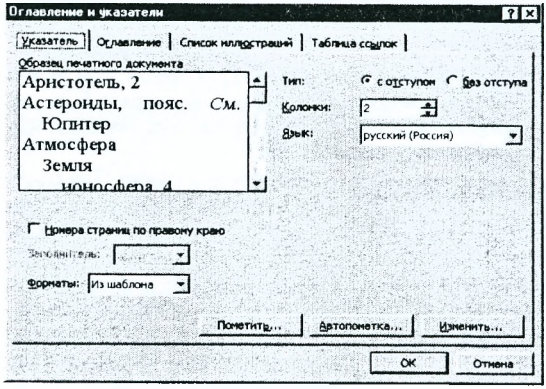


Рис. 5.25. Окно диалога для вставки Предметного указателя

Вставка оглавления

Оглавление является обязательным атрибутом любого более или менее сложного документа. В редакторе Word вставка оглавления предельно упрощена.

Для вставки оглавления выполните следующее:

- напишите заголовки и подзаголовки к тексту документа и оформите их с помощью стилей заголовков. Для этого выделите заголовок, откройте на панели инструментов Форматирование список стилей и выберите стиль заголовка соответствующего уровня (Рис. 5.26). Редактор позволяет создавать до 9 уровней заголовков (на практике редко используют больше трех уровней);

- перейдите в начало или конец документа и введите команду ***Вставка***, ***Оглавление и указатели*** и выберите в окне диалога закладку ***Оглавление*** (рис. 5.27);

- выберите формат, заполнитель, число уровней оглавления и щелкните по кнопке ОК.

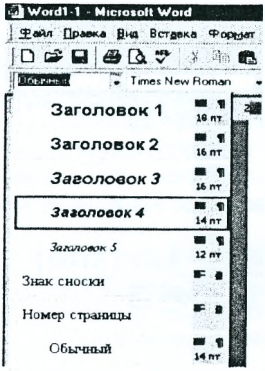


Рис. 5.26. Стили заголовков

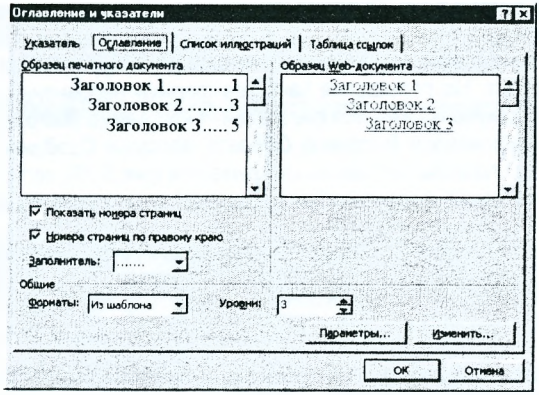


Рис. 5.27. Вставка Оглавления

Таким образом, единственное, что необходимо сделать, чтобы Word смог автоматически сформировать Оглавления документа, - оформить заголовки с помощью стилей заголовков.

При необходимости стиль заголовка можно изменить, это также делается очень просто:

- выделите заголовок, оформленный с помощью стиля заголовка, и установите требуемый тип, размер, начертание, цвет и другие параметры шрифта;

- активизируйте поле ввода списка Стиль на панели инструментов Форматирование щелчком мыши и нажмите кнопку Enter – открывается окно диалога Переопределение стиля (рис. 5.28);

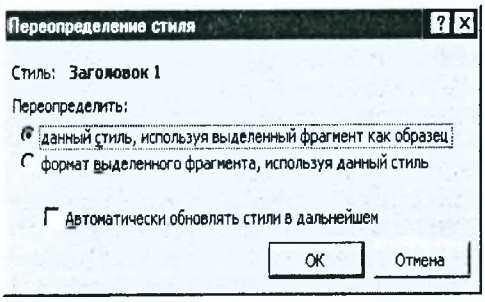


Рис. 5.28. Переопределение стиля заголовка

- установите переключатель Переопределить данный стиль, используя выделенный фрагмент как образец;

- установите флажок Автоматически обновлять стили в дальнейшем. В этом случае все заголовки, оформленные данным стилем, будут обновляться при переопределении стиля заголовка;

- щелкните по кнопке ОК для завершения работы.

Аналогичным образом можно создать собственный стиль для заголовка или абзаца, присвоив этому стилю имя, и использовать его в дальнейшем при оформлении документов.

Вставка формул

Часто при оформлении научных работ, отчетов требуется вписать в текст документа формулу. Раньше это делалось квалифицированными специалистами с помощью перьевой ручки и туши. Редактор Word предоставляет в распоряжение пользователя инструменты для выполнения этой работы самостоятельно пользователем. Нельзя сказать, что вписывание формул с помощью программ редактора быстрее, чем ручкой, скорее наоборот. Но повышение качества формул несомненно. Для вписывания формулы в текст документа имеется специальная программа ***Microsoft Equation 3.0***, которая вызывается командой ***Вставка, Объект***, закладка ***Создание***. При загрузке программы на экране появляется панель инструментов (рис.5.29), которая содержит набор элементов для вставки в формулы. При выборе любого значка открывается всплывающая панель с

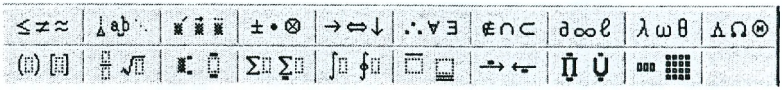


Рис. 5.29. Панель инструментов программы Microsoft Equation 3.0

набором значков, соответствующих выбранной теме. Каждый значок имеет знакоместа, куда вписываются символы. Выбор знакоместа осуществляется мышью.

$$Cos^3(x^2)\sqrt[3]{\frac{Lg(x)}{e^2}}+\sum_{\square}^{\square} \Rightarrow Cos^3(x^2)\sqrt[3]{\frac{Lg(x)}{e^2}}+\sum_{i=1}^{n} i^2$$

Вставка рисованных объектов

Редактор Word позволяет вставлять в текст документа картинки из набора рисунков или из файла, рисованные объекты, подготовленные средствами редактора или другими графическими редакторами, например, Paintbrush Picture, CorelDraw и др.

Для вставки картинки введите команду ***Вставка***, ***Рисунок*** – открывается всплывающее меню, которое позволяет вставить картинку, рисунок из файла, автофигуру, элемент WordArt, картинку со сканера или Диаграмму.

Технология вставки картинки следующая: выберите пункт меню ***Картинки*** – откроется окно диалога Вставка картинки. Откройте закладку Рисунки и выберите тему – открывается коллекция рисунков выбранной темы. Выберите рисунок – открывается панель инструментов управления просмотром и выбором рисунка и щелкните по кнопке Вставить клип (верхняя кнопка). Рисунок вставляется в документ. Чаще всего рисунок оказывается под окном диалога Вставка картинки и поэтому не виден пользователю. Чтобы увидеть рисунок, закройте окно диалога или сместите его в сторону.

Для вставки объекта из графического редактора Paintbrush выполните следующее: введите команду ***Вставка***, ***Объект*** и выберите на закладке "Создание" в списке "Тип объекта" объект Paintbrush Picture - откроется окно графического редактора. Создайте рисунок. Для вставки созданного рисунка в документ щелкните мышью по видимой части окна редактора Word. Чтобы отредактировать рисунок, щелкните по нему дважды мышью – откроется графический редактор.

Для вставки готового рисунка из файла имеется две возможности:

- введите команду ***Вставка, Рисунок*** и в выпадающем меню выберите команду ***Из файла*** – открывается окно диалога "Добавить рисунок". Можно вставлять в документ рисунки с расширением BMP, GIF, EMP, WMF, JPEG и многие другие;

- введите команду ***Вставка, Объект*** выберите закладку "***Создание из файла***", введите в строку ввода имя файла или щелкните по кнопке Обзор, чтобы выбрать файл из окна диалога.

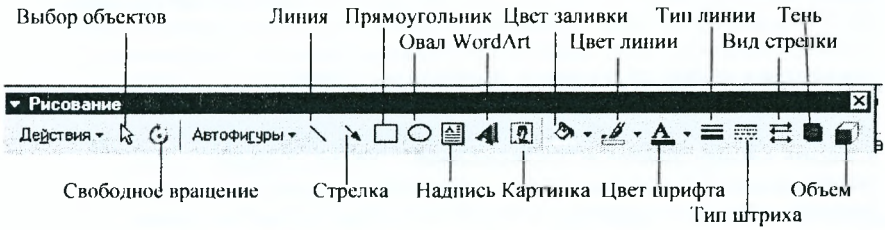


Рис. 5.30. Панель инструментов Рисование

Рисунки можно создавать и с помощью встроенных средств редактора. Откройте панель инструментов Рисование (рис. 5.30). На ней расположены инструменты для рисования и элементы управления.

Рассмотрим назначение некоторых элементов.

Инструменты для рисования: *Линия, Стрелка, Прямоугольник* и *Овал*, - служат для изображения соответствующих объектов. Для установки этих элементов на страницу документа выделите соответствующую кнопку мышью, переместите указатель мыши в нужную точку (указатель мыши меняет форму на тонкий черный крестик), нажмите левую клавишу мыши и протащите указатель мыши в нужном направлении. Примеры построенных объектов приведены на рис. 5.31. Чтобы выделить объект. щелкните по нему мышью. Выделенный объект отмечается маркерами в виде маленьких прямоугольников.

Рис. 5.31. Примеры объектов рисования

Для изменения размеров объекта зацепите мышью за маркер и протащите в нужном направлении. Все операции выполняются над выделенным объектом. Для выделенного объекта можно изменить цвет, тип линии, тип штриха с помощью соответствующих инструментов. Выделенный объект можно удалить, нажав клавишу *Del*. На линию можно установить стрелки с помощью инструмента *Вид стрелки*: выделите линию, щелкните по пиктограмме Вид стрелки - откроется коллекция стрелок. Выберите нужную. Чтобы нарисовать правильный квадрат или круг, нажмите и удерживайте во время рисования клавишу Shift. Прямоугольник и овал можно закрасить с помощью инструмента *Цвет заливки*, для нарисованного объекта можно изобразить тень, используя инструмент Тень, а с помощью инструмента Объем можно преобразовать плоскую фигуру в объемную. Инструменты *Цвет линии* и *Цвет заливки* имеют специальные команды *Нет линии* и *Нет заливки*. Эти команды используются, когда надо убрать линию рамки или цвет заливки. Инструмент *Надпись* позволяет построить

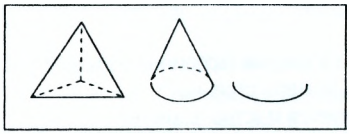


Рис.5.32. Примеры построения фигур

рамку, в которую можно помещать текст и другие объекты. Этот инструмент удобно использовать для изображения надписей к рисункам и другим объектам.

Используя инструменты рисования, можно изобразить любую фигуру, например, Пирамиду или Конус (рис. 5.32). Построить пирамиду легко: нарисуйте прямую линию, сделайте пять копий этой линии и соедините их концы, перетаскивая линии за маркеры выделения, невидимые линии сделайте пунктирными. Чтобы фигура не рассыпалась при перемещении ее надо сгруппировать: выделите инструмент Выбор объектов, установите указатель мыши выше левого верхнего края рисунка, нажмите клавишу мыши и обведите всю фигуру. Все объекты, полностью попавшие в контур выделения, выделяются. Выберите команду ***Действия, Группировать***. Объект становится одним целым. Для редактирования отдельных элементов рисунка объект надо разгруппировать командой ***Действия, Разгруппировать.*** Построить конус будет несколько сложнее. Для этого познакомимся еще с двумя инструментами панели Рисование: *Автофигуры* и *Свободное вращение*. Список "Автофигуры" содержит заготовки различных объектов, разбитых на группы по назначению. Например, в группе Основные фигуры есть инструмент *Дуга* (рис.5.32). Дуга при выделении, кроме восьми маркеров выделения, имеет еще два маркера в виде ромба желтого цвета. Маркеры желтого цвета позволяют изменять форму дуги, а другие маркеры – размеры. Для построения основания конуса нам нужно построить две дуги эллипса, одну - выпуклую вверх пунктирную, вторую - вогнутую вниз сплошную. Постройте первую дугу, она по умолчанию будет выпуклая вверх. Сделайте копию этой дуги и разверните ее на 180 градусов. Чтобы развернуть дугу, воспользуемся *инструментом Свободное вращение*. Выделите дугу и щелкните по инструменту *Свободное вращение* – возле дуги появится четыре круглых зеленых маркера. Зацепите мышью за один из маркеров и вращайте в нужном направлении. Соединив две дуги, получим основание конуса. Теперь остается добавить две образующих, сгруппировать рисунок и конус готов.

Список команд управления "Действия" содержит ряд важных команд для управления рисунками. Мы уже познакомились с командами Группировать и Разгруппировать. Познакомимся еще с несколькими.

Команда ***Порядок*** позволяет управлять положением объектов относительно друг друга и относительно текста.

Команда ***Обтекание текстом***, позволяет установить, как объект будет взаимодействовать с окружающим его текстом. Чаще всего применяют обтекание вокруг рамки и обтекание сверху и снизу. Например, на рис. 5.31. Установлен режим обтекания вокруг рамки. После установки режима обтекания объект можно легко переместить в нужное положение с помощью мыши: установите указатель мыши на объект – указатель принимает вид двунаправленных стрелок, нажмите клавишу мыши и перемещайте рисунок в нужное положение.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Изобразите лист формата А4 и покажите основные элементы оформления документа.
2. Как вставить в документ картинку, дату, сноску, номера страниц?
3. Как вставить в документ формулу? Как используется мастер формул?
4. Какие виды колонтитулов можно установить в документе? Как это выполнить?
5. Расскажите алгоритм вставки оглавления?
6. Расскажите назначение основных инструментов на панели инструментов Рисование?
7. Как нарисовать схему алгоритма в документе Word?

5.3. ТАБЛИЦЫ

Редактор Word имеет средства для работы с таблицами и обработки данных. Все команды для работы с таблицами сосредоточены в команде главного меню ***Таблица***. Для управления таблицами может использоваться также панель инструментов Таблицы и границы. Таблицы удобно применять для представления упорядоченного набора данных, списков, управления размещением текста, оформления логотипов и т. п.

Создание таблиц

Для создания таблицы введите команду ***Таблица, Добавить, Таблица***. В окне диалога укажите требуемое число строк и столбцов, щелкните по кнопке ОК. Другой удобный способ создания таблицы – щелкнуть по пиктограмме "Добавить таблицу" на панели инструментов Стандартная, выделить требуемое число строк и колонок путем протаскивания мышью по ячейкам. После отпускания клавиши мыши таблица появляется на текущем листе в месте вставки.

Таблица состоит из колонок и строк. По умолчанию все колонки нумеруются латинскими буквами от A до Z, а строки цифрами. Поэтому при написании формул к ячейке следует обращаться по ее адресу, например, A1.

Каждая ячейка таблицы представляет собой самостоятельный документ, в который можно помещать числа, текст, рисунки, даты, таблицы. Редактируется текст в ячейке по тем же правилам, что и в основном документе. Ширина колонок может изменяться путем перетаскивания границ ячеек с помощью мыши непосредственно на таблице или на горизонтальной линейке. Высота строки автоматически увеличивается при переносе текста, но может также изменяться путем перетаскивания линий разметки строк мышью.

Таблица становится активной, если щелкнуть мышью по любой ячейке. Для выделения строки подведите указатель мыши к левой ячейке снаружи таблицы так, чтобы он превратился в стрелку ↗, и щелкните мышью. Другой способ выделения строки – щелкните мышью по левой ячейке строки, нажмите клавишу мыши и протащите указатель

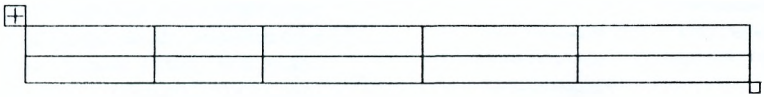


Рис. 5.33. Таблица с маркерами выделения

мыши по строке. Для выделения столбца подведите указатель мыши к верхней строке снаружи так, чтобы он превратился в черную стрелку ↓, и щелкните мышью. Выделить всю таблицу можно протаскиванием мыши или щелчком мыши по специальному маркеру, который появляется в верхнем левом углу таблицы при ее активизации (рис. 5.33). Подведите указатель мыши к границе строки или столбца внутри таблицы – появляется маркер выделения таблицы, подведите к нему указатель и щелкните мышью. Зацепив мышью за этот маркер, можно перемещать таблицу по листу.

Команды меню Таблица

Добавить – позволяет добавлять таблицу, столбец слева, столбец справа, строку сверху, строку снизу или ячейку. Вставка строки (столбца) осуществляется относительно выделенной строки (столбца). При вставке ячейки открывается дополнительное меню, которое предлагает указать направление смещения: вниз, вправо, вставить целую строку или целый столбец.

Удалить - позволяет удалить таблицу, строку, столбец или ячейку.

Выделить – позволяет выделить таблицу, текущую строку или текущий столбец.

Объединить ячейки и ***Разбить ячейки*** – позволяют объединять выделенные ячейки или разбивать их на несколько строк или столбцов.

<table>
<tr><td rowspan="4">1</td><td colspan="3">2</td><td rowspan="4">3</td></tr>
<tr><td rowspan="2">5</td><td colspan="2">6</td></tr>
<tr><td rowspan="2">7</td><td rowspan="2">8</td></tr>
<tr><td rowspan="2">4</td></tr>
</table>

Рис. 5.34. Пример объединения и разбиения ячеек

Таблица на рис. 5.34. имеет три столбца (1, 2, 3). Область 4 получена путем объединения двух ячеек первого столбца. Столбцы 5 и 6 получены путем деления четырех ячеек столбца 2 на два столбца, а столбцы 7 и 8 получены путем деления трех ячеек столбца 6 на два столбца.

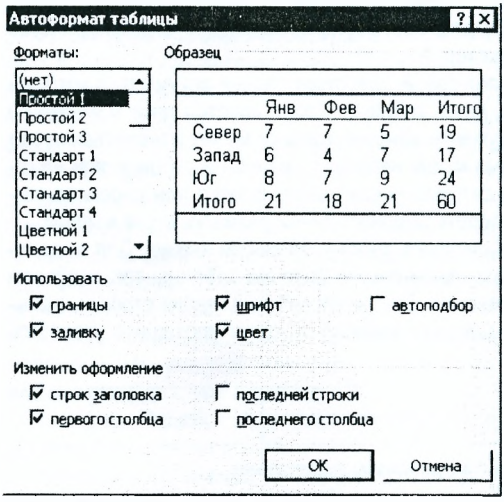


Рис. 5.35. Оформление таблицы с помощью команды Автоформат

Разбить таблицу – разбивает таблицу на две части по горизонтали по текущей строке, вставляя между ними пустую строку. Для объединения разбитых таблиц необходимо удалить пустые строки между ними. Эта команда работает корректно, если ширина таблицы равна ширине рабочей части листа.

Автоформат – выводит одноименное окно диалога, которое позволяет оформлять таблицу готовыми стилями. Имеется возможность редактирования выбранного стиля оформления таблицы путем установки или снятия нужных флажков (рис. 5.35). Для отмены оформления таблицы, выполненной с помощью команды Автоформат, выделите в списке "Форматы": окна диалога "Автоформат таблицы" значение "нет" (рис. 5.35) .

Автоподбор – позволяет автоматически настраивать ширину столбцов и высоту строк.

Заголовки – действует как переключатель. Позволяет установить/отменить режим автоматического формирования шапки таблицы при переходе текста таблицы на другую страницу. Выделите первую строку (строки) таблицы и введите команду ***Таблица, Заголовки.*** Как только текст таблицы перейдет на следующую страницу, автоматически сформируется шапка таблицы.

Преобразовать - позволяет преобразовывать таблицу в текст и наоборот – преобразовывать текст в таблицу. При преобразовании таблицы в текст, программа просит указать разделитель, которым будет отделяться текст: символ табуляции, точка с запятой, запятая или иной символ. При преобразовании текста в таблицу программа запра-

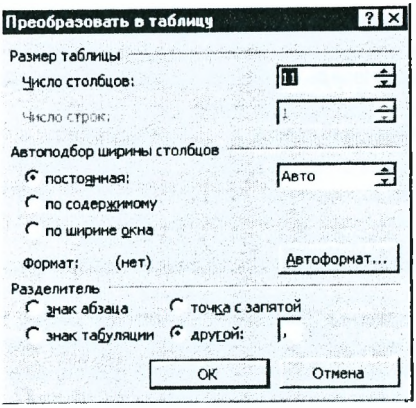


Рис. 5.36. Преобразование текста в таблицу

шивает, какой символ используется для разделения содержимого ячеек и просит также указать число столбцов. Поэтому текст, преобразуемый в таблицу, должен быть подготовлен определенным образом, а именно: необходимо расставить символы разделители. Например, имеется список птиц наших лесов и полей:

воробей, синица, зяблик, жаворонок, соловей, зимородок, кукушка, сорока, грач, галка, ворона.

Мы решили, что список будет читаться лучше, если он будет представлен в виде таблицы. Выделим получившийся текст и введем команду ***Таблица, Преобразовать в таблицу***. Открывается окно диалога "Преобразовать в таблицу" (рис. 5.36). Программа предлагает число столбцов 11 и разделитель - запятая. Установим число столбцов 3 и щелкнем по кнопке ОК – текст преобразуется в таблицу (рис. 37):

воробей	синица	зяблик
жаворонок	соловей	зимородок
кукушка	сорока	грач
галка	ворона	

Рис. 5.37. Текст, преобразованный в таблицу

Сортировка – позволяет отсортировать таблицу по возрастанию или убыванию значения. В качестве значения может быть текст, дата, число. Выделите таблицу, введите команду ***Таблица, Сортировка*** – откроется одноименное окно диалога (рис.5.38).

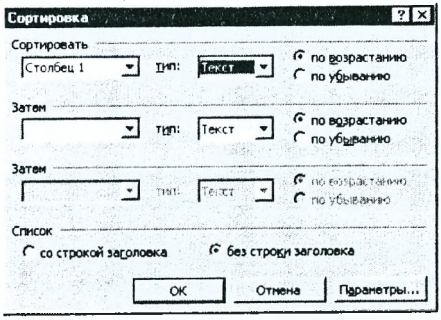


Рис.5.38. Сортировка списков

Окно диалога позволяет задать одновременно три условия сортировки. Укажите, какой столбец будет использоваться первым, вторым и третьим, тип данных в каждом столбце и направление сортировки (по возрастанию или убыванию значения). При задании последовательности сортировки необходимо исходить из следующего принципа: первый параметр должен быть наиболее общим, то есть он должен разбивать всю совокупность данных на наименьшее число групп.

Пусть, имеется список рабочих (рис.5.39). Определить, в какой последовательности следует выбирать поля таблицы для сортировки, если:

а) требуется отсортировать таблицу по трем полям: Фамилия и инициалы, Номер цеха, Трудовой стаж;

б) требуется составить список рабочих по цехам.

Для пункта а) первым параметром для сортировки следует выбрать Номер цеха,

вторым параметром можно выбрать трудовой стаж, а третьим фамилию. В этом случае программа распределит рабочих сначала по цехам, потом в переделах цехов распределит рабочих по трудовому стажу и в последнюю очередь в пределах этих групп распределит их в алфавитном порядке.

Фамилия и инициалы	НОМЕР ЦЕХА	*Табельный номер*	*Трудовой стаж*
Петров И. С.	1	А342	5
...	8	4	

Рис. 5.39. Пример таблицы для определения порядка выбора полей для сортировки

Для пункта б) в первую очередь необходимо отсортировать таблицу по номеру цеха, а затем по фамилиям и инициалам.

Формула – таблица позволяет проводить несложные вычисления. Выделите ячейку, в которую будет вводиться формула и введите команду Таблица, Формула. Отроется окно диалога (рис. 5.40). Строка Формула служит для ввода формулы. Формула должна начинаться со знака равно и может содержать ссылки на ячейки таблицы и функции. Например: A1*B1+C1*D1, PRODUCT(A1:C1), AVERAGE(C1:C10) и т. д. В функциях в качестве аргументов могут указываться адреса ячеек, списки адресов ячеек, указанные через символ ";" или диапазоны. Диапазон – непрерывная область ячеек, обозначенная номером первой верхней и последней нижней ячеек группы, разделенных двоеточием. Например,

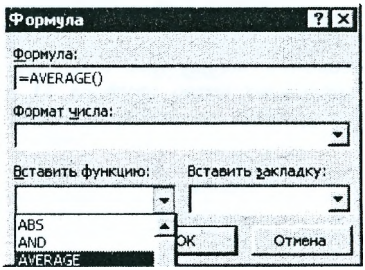


Рис. 5.40. Ввод формул

SUM(A1;C1;E1) – вычисляется сумма значений ячеек A1, C1 и E1;

SUM(A1:D8) – вычисляется сумма значений ячеек, расположенных в блоке (диапазоне) ячеек A1: D8.

Список Вставить функцию содержит ряд встроенных арифметических и логических функций. Например:

ABS() – абсолютная величина числа;

AVERAGE() – среднее значение списка или диапазона чисел;

COUNT() – количество значений списка или диапазона чисел;

MAX()– максимальное значение в списке или диапазоне чисел;

MIN() – минимальное значение в списке или диапазоне чисел;

PRODUCT() – произведение значений в списке или диапазоне.

В качестве аргумента у встроенных функций могут использоваться также служебные слова LEFT - список всех ячеек слева, ABOVE – список всех значений выше, например, SUM(ABOVE).

Скрыть сетку – эта команда позволяет управлять отображением сетки при печати. Команда действует как переключатель и имеет два значения: скрыть сетку и отображать сетку.

Свойства – эта команда позволяет управлять размещением таблицы на листе, настраивать параметры таблицы, строк, столбцов и ячеек.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначены таблицы в редакторе Word?
2. Как создать таблицу?
3. Как добавить строку, столбец, ячейку?
4. Как настроить свойства таблицы, строки, столбца?
5. Как осуществляются вычисления в таблице?

5.4. ШАБЛОНЫ

Основой каждого документа является шаблон. Шаблон – это набор параметров форматирования текста, абзацев, списков, элементов автотекста, макросов.

Редактор Word имеет стандартный шаблон Normal, который загружается при открытии нового документа. Кроме того он имеет большое количество других шаблонов для создания записок, факсов, писем, публикаций и т. п. Чтобы найти требуемый шаблон, введите команду Создать, открывается окно диалога (рис. 5.41). Выберите подходящую по содержанию закладку, выберите в списке шаблонов нужный шаблон и щелкните по

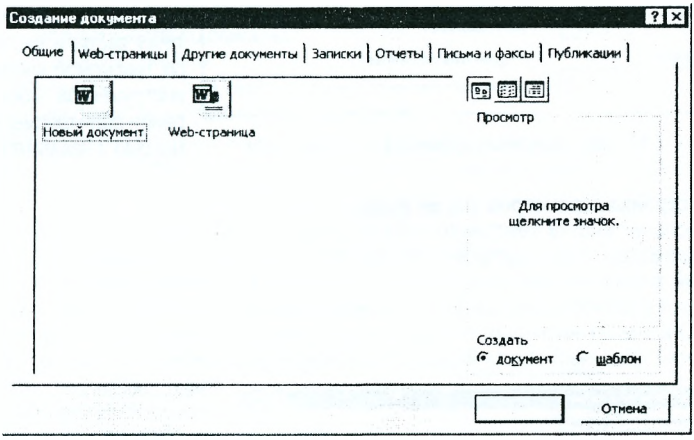


Рис. 5.41. Окно диалога Создание документа

кнопке ОК. Шаблон может содержать поля ввода с текстом, который нужно заменить на свой, например, название организации, домашний адрес и т. п.

В документ можно вставлять поля различного вида с помощью команды Вставка, Поле или панели инструментов Форма. Ограничимся рассмотрением тех полей, которые расположены на панели управления Форма.

Панель управления Форма вызывается командой ***Вид, Панели управления***, установить флажок ***Формы*** (рис. 5.42). Поля формы панели имеют следующее значение:

Рис. 5.42. Панель управления Формы

- *текстовое поле* служит для вставки текста, чисел, дат и времени;
- *флажок* – предназначен для вставки элементов управления Флажок. Флажок может иметь два состояния: установлен или снят;
- *список* служит для ввода данных из списка;
- *параметры поля формы* служат для настройки свойств полей формы;
- *затенение* – предназначено для выделения полей формы серым цветом;

- *защита* – обеспечивает защиту постоянной информации от изменения. После установки защиты курсор будет перемещаться только по полям формы.

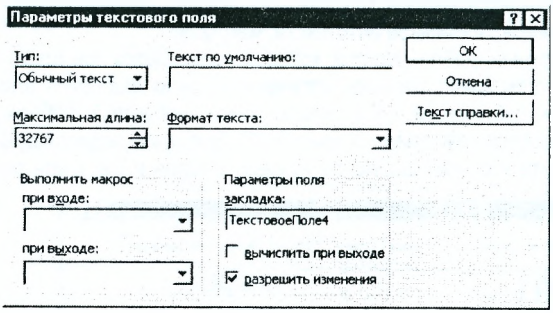


Рис. 5.43. Окно настройки параметров текстового поля

Другие элементы управления на панели инструментов Формы не являются полями формы.

Для установки поля формы в документ установите курсор в точку ввода и щелкните мышью по требуемой кнопке на панели инструментов Формы. Для удаления поля формы выделите его мышью и нажмите клавишу Del.

Настройка параметров полей формы

После установки текстового поля в документ нужно, при необходимости, настроить его параметры. Окно настройки параметров вызывается щелчком мыши по соответствующей кнопке на панели инструментов, двойным щелчком мыши по полю формы или через контекстное меню. Окно настройки параметров текстового поля приведено на рис. 5.43. Оно имеет несколько списков и строку ввода. Список "Тип" позволяет установить тип поля: обычный текст, число, дата, время или вычисления. Счетчик "Максимальная длина" позволяет установить ограничения на длину поля формы. Список "Формат" позволяет устанавливать формат текста, даты, времени или числа. Строка ввода "Текст по умолчанию" позволяет вводить, в зависимости от установленного типа, текст, число, дату или формулу. Для каждого поля формы автоматически формируется закладка.

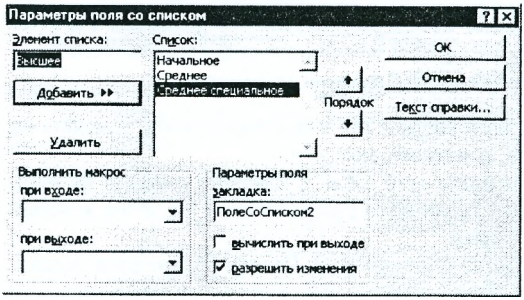


Рис. 5.44. Окно настройки параметров поля со списком

Окно настройки параметров поля со списком позволяет вводить данные в список (рис.5.44), а также изменять порядок следования элементов списка с помощью кнопок Порядок. При настройке параметров флажка можно установить флажок на умолчание или снять его.

Пример создания шаблона

Рассмотрим порядок создания собственного шаблона на примере Анкеты участника соревнований (условный пример).

Предположим, что нам нужно вести учет спортсменов, прибывающих на соревнование по легкой атлетике. При этом требуется отразить следующие сведения: фамилия, имя, отчество, домашний адрес, номер телефона, дата рождения, возрастная группа, вид соревнования, семейное положение, дистанция, личные достижения, является ли обладателем рекордов: мирового, европейского, республиканского.

Разработаем форму анкеты (рис. 5.45).

Порядок работы:

1. Введите команду Файл, Создать. В окне диалога Создание документа (рис. 5.41) щелкните по кнопке Новый документ, установите переключатель Шаблон и щелкните по кнопке ОК.

2. Введите неизменяемую часть информации (Заголовки, таблицу, текст, рисунки и др.).

3. Вставьте поля формы и настройте их свойства:

по умолчанию Стадион "Динамо";

- Дата – поле типа Дата, длина 15, с шаблоном и примером заполнения;
- Время – текстовое поле, длина 10, заполнено по умолчанию;
- Возрастная группа – поле со списком;
- Вид соревнований – поле со списком;
- Дистанция – поле со списком;
- Личное достижение - простой текст, длина 10;
- Фамилия, имя, отчество – простой текст, длина 40;
- Место проведения соревнований – текстовое поле, длина поля 20,текст
- Дата рождения – поле типа Дата с шаблоном ДД/ММ/ГГ, длина 8, пример заполнения;
- Адрес, Город – простой текст, длина поля –50;
- Страна – простой текст, длина 30, текст по умолчанию;

Анкета участника соревнований

Место проведения соревнований:

Дата: Время:

Возрастная группа →			
Вид соревнования	Дистанция	Личное достижение	
Фамилия, имя, отчество			
Дата рождения			
Адрес			
Город		Индекс →	
Страна		Телефон →	
Дополнительные сведения являетесь ли обладателем рекордов: мирового ☐ европейского ☐ республиканского ☐			

Рис. 5.45. Пример шаблона до настройки свойств полей

Анкета участника соревнований

Место проведения соревнований: Стадион "Динамо"

Дата: 08/06/04 Время: 10 часов

Возрастная группа → общая (до 21)			
Вид соревнования Бег	Дистанция 10 км	Личное достижение 27,35 мин	
Фамилия, имя, отчество	Иванов Петр Михайлович		
Дата рождения	12/05/1984		
Адрес	ул. Запрудная дом 40, кв. 17		
Город	Бобруйск	Индекс → 224000	
Страна	Белоруссия	Телефон → 23-15-67	
Дополнительные сведения являетесь ли обладателем рекордов: мирового ☐ европейского ☐ республиканского ☒			

Рис. 5.46. Пример шаблона после настройки полей и установки защиты

- Почтовый индекс – простой текст с примером заполнения;
- Телефон – число, длина 8, пример заполнения;
- сведения об обладании рекордом – флажки, не установлены

4. Установите защиту.

5. Сохраните шаблон командой Файл, Сохранить, введите имя файла.

Файл сохраняется по умолчанию с расширением .dot в папке Шаблоны.

Пример шаблона после настройки полей и установки защиты приведен на рис. 5.46.

Использование шаблона

Шаблон после сохранения попадает в список документов на закладке "Общие" окна диалога "Создание документа". Для использования шаблона введите команду ***Файл, Создать***. Выделите в окне диалога "Создание документа" на закладке "Общие" созданный вами шаблон, установите переключатель Документ и щелкните по кнопке ОК. После этого заполните поля формы. Сохраните документ на диске и, при необходимости, напечатайте.

Для редактирования шаблона введите команду ***Файл, Создать***, выделите в окне диалога "Создание документа" шаблон, установите переключатель Шаблон и щелкните по кнопке ОК. Вызовите панель инструментов Формы и снимите защиту. Теперь можно приступать к редактированию шаблона.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое шаблон, для какой цели он создается?
2. Расскажите алгоритм создания шаблона?
3. Какие поля можно использовать в шаблоне и как они вставляются?
4. Как настроить свойства (параметры) полей формы?

5.5. СЛИЯНИЕ ДОКУМЕНТОВ

Подготовка документов к рассылке

Часто приходится рассылать один и тот же документ по разным адресам, например, приглашение на презентацию книги, юбилей учебного заведения, симпозиум и др. Программа Word позволяет автоматизировать эту работу. Для подготовки к рассылке документа, оформления конвертов и наклеек имеется *Мастер слияния документов*.

Для выполнения операции слияния необходимо подготовить шаблон документа с полями слияния и источник данных для заполнения этих полей.

Шаблон документа может быть подготовлен заранее и сохранен на диске как документ Word или создан в процессе выполнения работы.

Обращение	Имя	Фамилия	Должность	Организация	Адрес1	Адрес2	Город	Область	Индекс

Рис. 5.47. Пример источника данных для слияния

Источник данных представляет собой обычную таблицу. Таблица может быть подготовлена средствами Word или другими приложениями, например, с помощью электронной таблицы, системы управления базой данных и др. Пример шапки таблицы приведен на рис. 5.47.

Объединение источника данных с шаблоном документа осуществляется с помощью Мастера слияния документов.

Мастер слияния документов вызывается командой ***Сервис, Слияние.*** На экране появляется окно диалога *Слияние* (рис. 5.48). Оно содержит три команды: Основной документ, Источник данных и Объединение.

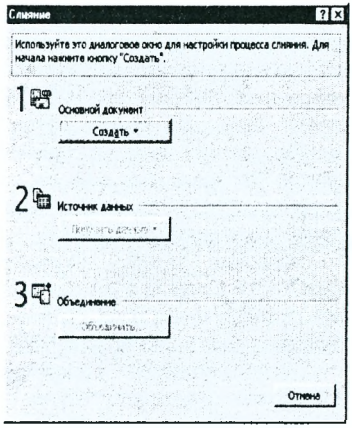


Рис.5.48. Окно диалога мастера слияния

Команда ***Основной документ*** содержит список "Создать", который позволяет создать документ на бланке, конверты, наклейки, каталог или преобразовать в обычный документ. После выбора пункта меню открывается следующее окно, которое предлагает использовать в качестве основного документа текущий документ (активное окно) или создать новый документ (рис. 5.49). После выбора варианта создания основного документа документ сохраняется на диске и активизируется вторая команда Источник данных.

Команда ***Источник данных*** позволяет создать Новый источник данных, Открыть источник данных, Использовать адресную книгу или Параметры заголовков. Команда *Новый источник данных* позволяет создать новую базу данных. Команда *Открыть источник данных* позволяет открыть существующую базу данных. Команда *Использовать адресную книгу* позволяет загрузить источник данных подготовленный другими приложениями Windows. Команда *Параметры заголовка* позволяет загрузить источник данных для заголовков таблицы источника данных, если данные и заголовки хранятся в разных файлах.

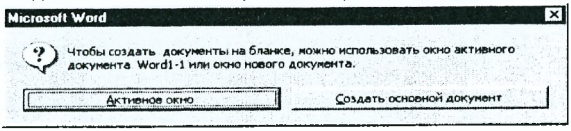


Рис. 5.49. Выбор варианта создания основного документа

Команда Новый источник данных открывает окно диалога Создание источника данных (рис. 5.50). Список "Поля в строке заголовка" уже содержит список полей базы данных. С помощью строки ввода "Поле" и кнопки Добавить поле можно добавить новые поля в список, а с помощью кнопки Удалить поле – удалить лишние поля. Имя поля базы данных не должно содержать пробелов. Кнопки Порядок позволяют изменить порядок следования полей в базе данных. Кнопка MS Query позволяет сформировать запрос для загрузки имен полей из другой таблицы.

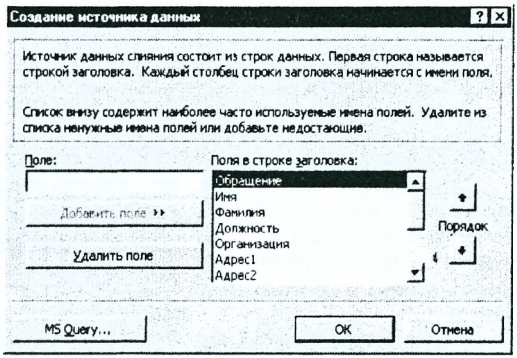


Рис. 5.50. Создание источника данных

После щелчка по кнопке ОК программа предлагает сохранить данные на диске, а затем выполнить правку источника данных или основного документа. Если база данных еще не создана, то следует ввести команду *Правка источника данных* – на экран выводится окно диалога для ввода данных в базу данных (рис. 5.51). Заполните поля базы данных. После щелчка по кнопке ОК программа предлагает перейти к правке основного документа.

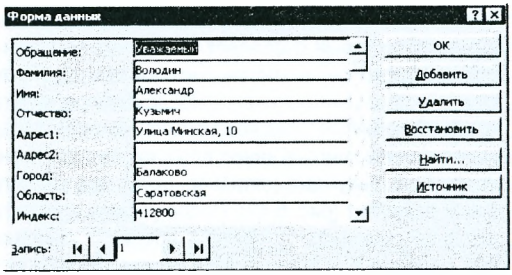


Рис. 5.51. Заполнение базы данных

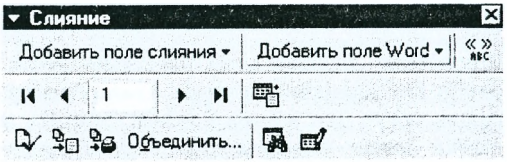


Рис. 5.52. Панель инструментов Слияние

В окне основного документа, кроме известных панелей инструментов, появится новая панель инструментов Слияние (рис. 5.52). В левой части этой панели инструментов имеется раскрывающийся список "Добавить поле слияния", в котором содержится список всех полей созданной базы данных.

Напишите текст документа (рис.5.53) и вставьте в нужных местах щелчком мыши поля из списка "Добавить поле слияния" (рис. 5.52).

Теперь остается выполнить последнюю операцию: передать значения полей из базы данных в шаблон документа. Эта операция выполняется командой *Объединить* на панели инструментов

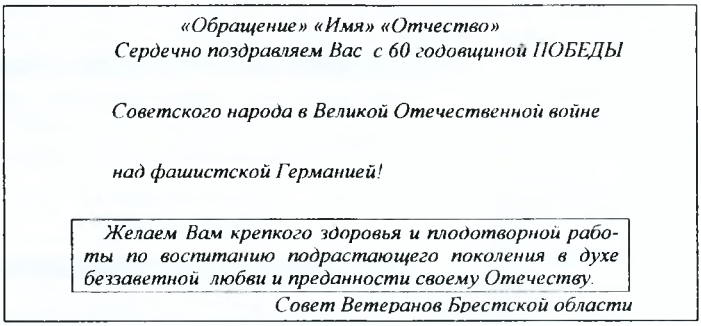


Рис. 5.53. Шаблон основного документа с полями слияния

Слияние или аналогичной командой в окне диалога Слияние.

Окно диалога Слияние при выполнении команды Объединить приведено на рис. 5.54. Оно позволяет управлять отбором данных и печатью документа.

Кнопка Отбор записей позволяет устанавливать критерии отбора. Критерии отбора могут объединяться по схеме И или ИЛИ. Критерии должны быть не противоречивыми. Можно установить до пяти кри-

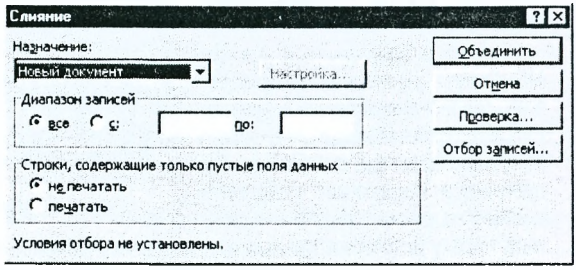


Рис.5.54. Окно диалога Слияние при объединении

териев отбора, например по названию организации, городу, фамилии или обращению, как показано на рис. 5.55. Вторая закладка этого окна диалога позволяет сортировать записи базы данных. Можно использовать до трех полей базы данных для сортировки (см. раздел 5.6. Таблицы).

Создание конвертов

После создания документов их необходимо разложить по конвертам, надписать адреса и разослать по почте. Понятно, что это тоже трудоемкая работа. Поэтому целесообразно поручить ее мастеру слияния документов.

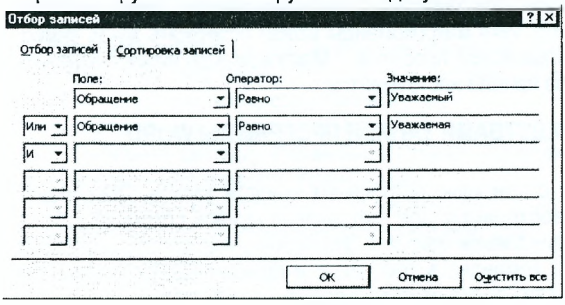


Рис. 5.55. Отбор записей при объединении

При подготовке конвертов может использоваться та же информационная база, которая использовалась при создании документов.

Сначала необходимо записать адрес отправителя. Введите команду ***Сервис, Параметры.*** Откройте закладку ***Пользователь*** и в окне Почтовый адрес запишите юридический адрес своей организации (свой домашний адрес) по правилам отправки корреспонденции.

Теперь можно приступить к подготовке конвертов:

- откройте новый документ;
- введите команду ***Файл, Параметры страницы***. На закладке Размер бумаги выберите формат бумаги, соответствующий размеру конверта;
- введите команду ***Сервис, Слияние***;
- введите команду ***Основной документ, Создать, Конверты, Активное окно***;
- введите команду ***Источник данных, Получить данные, Открыть источник данных.*** Выберите файл базы данных и на запрос программы введите команду ***Настройка основного документа***;
- в окне диалога Параметры конверта установите размеры конверта и настройте параметры печати;
- в следующем окне диалога - Адрес на конверте вставьте в окне "Образец адреса" поля слияния для адреса получателя, выбирая их из списка "Вставить поле слияния";
- введите команду ***Объединить.*** Установите, при необходимости, параметры отбора в окне диалога Слияние и завершите работу щелчком по кнопке Объединить.

Надписи для конвертов готовы. Теперь их можно просмотреть и напечатать.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего используется мастер слияния документов?
2. Что такое основной документ, как его создать?
3. Что такое источник (база) данных, как он создается?
4. Расскажите порядок слияния документов?
5. Как осуществляется выбор адресатов при рассылке писем?
6. Где необходимо указать свой домашний адрес для автоматического переноса его в адрес на конверте?
7. Как заадресовать конверты для рассылки писем?

5.6. МАКРОКОМАНДЫ

Макрокоманды или иначе Макросы позволяют легко и эффективно описывать часто повторяющиеся действия и тем самым повышать производительность туда пользователя.

Макрокоманды хранятся в шаблонах. Выбором шаблона для сохранения ограничивается круг документов, в которых можно использовать ту или иную макрокоманду. Место хранения макроса указывается при его создании. Макрокоманду можно также скопировать из одного шаблона в другой, используя команду Организатор из меню Формат, Стиль.

Макрокоманда может быть написана пользователем с использованием встроенного языка программирования Visual Basic или записана с помощью средств записи программы Word. Макрокоманда имеет имя. Имя макрокоманды может содержать до 36 символов, в имени не допускается использование пробелов. Макрокоманде можно назначить комбинацию клавиш или кнопку на панели инструментов.

5.6.1. ЗАПИСЬ МАКРОСА СРЕДСТВАМИ ЗАПИСИ ПРОГРАММЫ WORD

Запись макроса

Проще всего создать макрос с помощью встроенных средств записи. Продумайте последовательность действий, необходимых для выполнения нужной операции (сценарий), а затем повторите их в режиме записи макроса. Это необходимо делать всегда, так как в макрос будут записаны все команды, в том числе и ошибочные. Создадим для примера макрос для удаления текста справа от точки вставки.

Требуемая последовательность действий: установить курсор в строку текста слева от удаляемого текста; выделить текст, нажав комбинацию клавиш Shift + End; нажать клавишу Delete.

Начать записать макроса можно двумя способами: кнопкой ***ЗАП*** в строке состояния или командой ***Начать запись***: Сервис, Макрос, Начать запись.

Алгоритм записи макроса:

- установите курсор в любое место в тексте;
- щелкните дважды мышью по кнопке ЗАП в строке состояния. Открывается окно диалога Запись макроса (рис. 5.56);

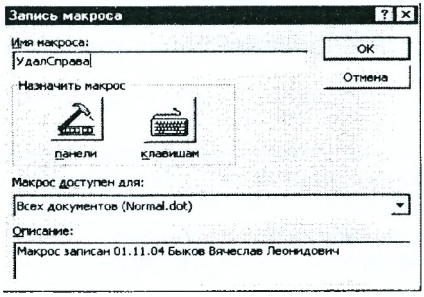


Рис. 5.56. Окно диалога записи макроса

- введите имя макроса в строке ввода Имя макроса. Выберите в списке "Макрос доступен для" документ, где будет храниться макрос. В окне Описание дайте, при необходимости, описание макроса или оставьте текст, сформированный программой по умолчанию. Текст описания не должен превышать 255 символов. При вводе длинного текста для перехода на новую строку нажмите Shift + Enter. Макросу можно назначить кнопку на панели инструментов или комбинацию клавиш с помощью кнопок раздела "Назначить макрос". После выполнения всех необходимых операций щелкните по кнопке ОК. Появляется окно диалога управления записью (рис. 5.57) . Это окно имеет две кнопки Остановить запись и Пауза/Продолжить запись. С этого момента все ваши действия будут записаны в макрос;

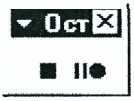


Рис. 5.57

- выполните последовательность действий для удаления текста справа согласно разработанному ранее сценарию;

- щелкните по кнопке ***Остановить запись*** на панели диалога или введите команду ***Сервис, Макрос, Остановить запись***.

Выполнение макроса

Для запуска макроса установите курсор слева от текста, который надо удалить, и введите команду Сервис, Макрос, Макросы, выберите в списке нужный макрос и щелкните по кнопке Выполнить.

Процесс применения макроса значительно упрощается, если назначить макросу комбинацию клавиш или назначить кнопку на панели инструментов.

Назначение макроса кнопке на панели инструментов

Назначить макрос кнопке панели инструментов, пункту меню или назначить макросу комбинацию клавиш можно сразу же при создании макроса в окне диалога Запись макроса (рис. 5.56) или после создания макроса.

Введите команду ***Сервис, Настройка***. Открывается окно диалога Настройка. (рис. 5.58). Выберите закладку ***Команды***. В списке "Категории" выберите ***Макросы***, а в списке Команды – имя макроса ***УдалСправа***.

Зацепите имя макроса мышью и перетащите его на одну из панелей инструментов – на панели инструментов появляется новая кнопка с именем макроса (можно создать новую панель инструментов: выберите закладку Панели инструментов и введите команду Создать).

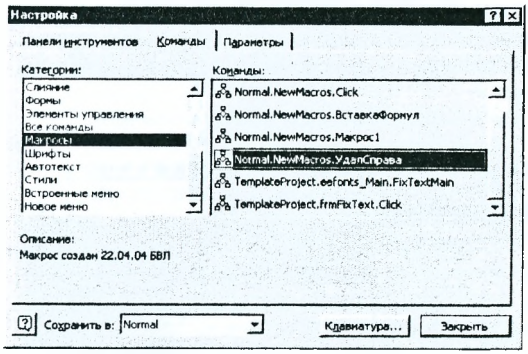


Рис. 5. 58. Окно диалога настройка макроса

Если вид кнопки с длинной надписью не устраивает Вас, то ее можно отредактировать. Выделите новую кнопку – в окне диалога Настройка появляется кнопка Изменить выделенный объект. Щелкните по этой кнопке. Открывается всплывающее меню. Выделите в этом меню последовательно команды Выбрать значок, Основной стиль. Команда Выбрать значок открывает библиотеку заготовок, из которой можно выбрать подходящий значок, например, ➔. Команда Основной стиль оставляет на кнопке только рисунок. Замените текст на более подходящий или оставьте его без изменения.

Назначение макросу комбинации клавиш

Введите команду Сервис, Настройка. В окне диалога Настройка (рис. 5.58, 5.56) щелкните по кнопке Клавиатура – откроется окно диалога Настройка клавиатуры (рис.5.59). В списке Категории выберите команду Макросы, а в списке Макросы – имя

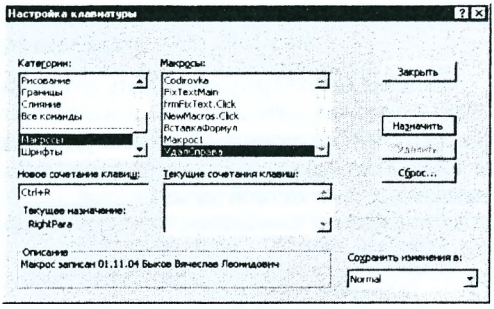


Рис. 5.59. Назначение комбинации клавиш макросу

макроса УдалСправа. Активизируйте строку "Новое сочетание клавиш" щелчком мыши и нажмите требуемую комбинацию клавиш, например, Ctrl+R. Если ранее макросу было назначено какое-то сочетание клавиш, то оно будет отображено в окне Текущее сочетание клавиш. В списке "Сохранить изменения в ..." укажите имя шаблона, где будет храниться макрос. Завершите работу щелчком по кнопке ***Закрыть***. Теперь для запуска макроса достаточно установить курсор слева от удаляемого текста и нажать комбинацию клавиш Ctrl+R.

5.6.2. ЗАПИСЬ МАКРОСА С ПОМОЩЬЮ ВСТРОЕННОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ VISUAL BASIC

Для записи макроса с помощью встроенного языка программирования Visual Basic введите команду ***Сервис, Макрос, Редактор Visual Basic*** – открывается окно разработки проекта (рис. 5.60).

Введите команду ***Insert, Module*** – открывается окно разработки программы Module1. Введите команду Insert, Procedure – открывается окно добавления процедуры Add Procedure. Введите имя процедуры, например, Codirovca, установите переключатели Sub – ключевое слово заголовка процедуры и Public – общая (процедура будет доступна всем формам проекта) и щелкните по кнопке OK – программа возвращается в окно проекта Module1. В этом окне появится шаблон процедуры - две строки команд, между которыми необходимо записать текст программы:

```
Public Sub Codirovca()
End Sub
```

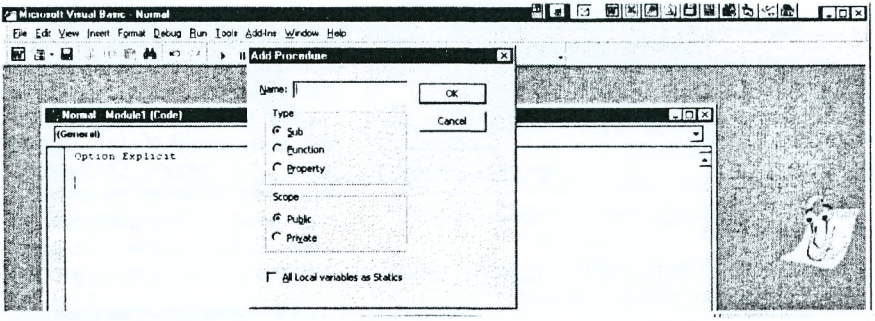


Рис. 5.60. Окно разработки программы Visual Basic

Если Вы умеете программировать или позаимствовали текст понравившейся процедуры у товарища, напишите здесь или скопируйте сюда текст программы. Часто бывает, что мы забываем переключить клавиатуру с русского на английский текст или наоборот, и тогда досадуем на себя за такую оплошность. Проблему легко устранить с помощью программы, текст которой приведен ниже.

```
Rem Программа перекодировки текста с русского на английский и наоборот
Public Sub Codirovka()   ' заголовок процедуры
  On Error Resume Next   ' обработчик ошибок
  ' Объявление переменных:
  Dim n As Integer, strSetRus As String, strSetEng As String
  Dim strMisStr  As String, strCurrChar As String
  Dim strNewStr As String, numChrPos As Integer
  ' strMisStr – пременная для хранения  ошибочного текста
```

```
    strMisStr = Selection.Text ' присвоение выделенной строки переменной
    ' strSetEng – строка, содержащая символы английского языка
    strSetEng = "QWERTYUIOP{}ASDFGHJKL:ZXCVBNM<>qwertyuiop[]asdfghjkl; zxcvbnm,."
    '  strSetRus - строка, содержащая символы русского алфавита
    ' русские и английские символы записаны не в алфавитном порядке,
    ' а в соответствии с раскладкой клавиатуры
   strSetRus="ЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЯЧСМИТЬБЮйцукенгшщзхъфы
вапролджячсмитьбю"
  ' цикл: извлекается символ из строки с ошибочным текстом и сравнивается со
  ' строками, содержащими английский и русский текст. Если в ошибочной
  ' строке английский текст, то он будет заменен на русский и наоборот
  ' исправленный текст помещается в переменную strNewStr
  For n = 1 To Len(strMisStr)
       strCurrChar = Mid(strMisStr, n, 1)
       numChrPos = InStr(strSetEng, strCurrChar)
       If numChrPos <> 0 Then
          strCurrChar = Mid(strSetRus, numChrPos, 1)
       Else
          numChrPos = InStr(strSetRus, strCurrChar)
          strCurrChar = Mid(strSetEng, numChrPos, 1)
       End If
       strNewStr = strNewStr & strCurrChar
    Next n
    '  выделенному тексту присваивается новая строка
    Selection.Text = strNewStr
End Sub
```

Когда программа будет написана, сохраните ее на диске командой ***File, Save Normal***, а затем вернитесь в документ командой ***File, Close and Return to Microsoft Word***. Наша программа попала в список макросов.

Чтобы применить ее, выделите ошибочный текст и введите команду ***Сервис, Макрос, Макросы***. Выделите макрос ***Codirovka*** и щелкните по кнопке Выполнить. Для удобства использования назначьте макрос комбинации клавиш или кнопке панели инструментов, как описано выше в данном разделе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое "макрос". Как его создать?
2. Как используются макросы?
3. Как назначить макрос кнопке панели инструментов?
4. Как назначить макросу комбинацию клавиш?
5. Как написать макрос с помощью встроенного языка программирования?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с текстовым процессором Microsoft Word. Он обладает большими возможностями по редактированию текста, размещения текста в виде колонок, создания списков, бюллетеней, вставки рисунков и рисованных объектов, формул. Имеется возможность создавать таблицы и управлять ими, проводить несложные вычисления. Редактор позволяет создавать базы данных и использовать их для рассылки документов, писем. Имеется также возможность автоматизации часто повторяющихся операций путем создания макросов. В разделе рассмотрены далеко не все возможности процессора, что ограничено рамками пособия.

6. ЭЛЕКТРОННАЯ ТАБЛИЦА EXCEL

6.1. ОСНОВНЫЕ СВЕДЕНИЯ

Электронная таблица Excel - интегрированная система. Она предназначена для создания и обработки электронных таблиц, списков (баз данных), представления результатов обработки таблиц и списков в виде диаграмм и графиков функций, подготовки выходных форм документов, сохранения их на дисках и вывода на печать.

Для загрузки программы запустите программу Windows и щелкните мышкой по значку приложения Microsoft Excel в панели инструментов Microsoft Office. Для выхода из программы введите команду ***Файл***, ***Выход***.

Описание рабочего окна Excel

В верхней части окна расположена строка заголовка, в которой располагаются кнопка системного меню, название приложения и имя файла, загруженного в окно (в начале работы по умолчанию выводится имя файла, предлагаемого по умолчанию - Книга-1), кнопки свертывания и развертывания окна программы. Ниже расположены: строка меню, стандартная панель инструментов, панель инструментов форматирования, строка ввода данных, рабочее окно (рис. 6.1).

Строка меню обеспечивает доступ ко всем командам электронной таблицы.

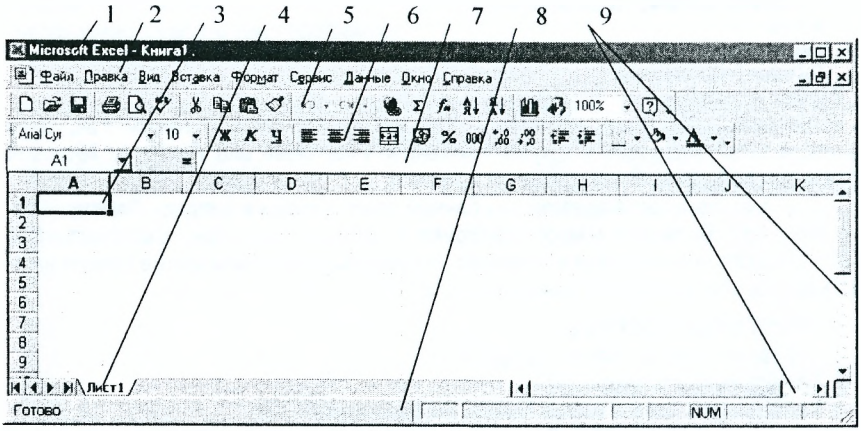


Рис. 6.1. Рабочее окно программы Excel: 1- строка заголовка, 2 – меню, 3 – активная ячейка, 4 – ярлычок листа, 5 панель инструментов стандартная, 6 – панель инструментов форматирования, 7 – строка формул, 8 – строка состояния, 9 – линейки прокрутки,

Стандартная панель инструментов содержит кнопки, дублирующие основные команды меню. Щелкнув мышью по кнопке панели, можно сразу же вызвать нужную команду. Назначение кнопок высвечивается при зависании указателя мыши на соответствующей кнопке.

Панель Форматирования позволяет оформлять рабочий лист путем выбора типа данных, параметров шрифта, размещения содержимого в ячейках, обрамления ячеек, а также выбора цвета шрифта, линий и фона.

Панели инструментов могут убираться или дополняться с помощью команды ***Вид***,

Панели инструментов. У панелей, выведенных на экран, установлены флажки. Для вывода нужной панели на экран установите флажок. Для удаления панели с экрана снимите флажок, щелкнув по соответствующему флажку мышью.

Строка формул имеет три поля: поле адреса ячейки, поле управляющих клавиш и поле ввода данных. В поле *Адреса ячейки* указан адрес текущей ячейки или ее имя. Если щелкнуть мышкой по этому полю, ввести адрес ячейки и нажать клавишу Enter, то курсор электронной таблицы перейдет в указанную ячейку. Этот прием перехода к нужной ячейке называется *непосредственной адресацией.* В *поле управляющих кнопок* выводятся три кнопки: X - отмена редактирования строки ввода, ✓ - окончание редактирования, "=" – ввод символа "=". Названные кнопки появляются при активизации строки ввода. Поле *Ввода данных* предназначено для отображения вводимой информации или содержания выделенной ячейки. Для активизации строки ввода необходимо щелкнуть по ней мышью.

Строка состояния расположена в нижней части рабочего окна. В левой части строки состояния выводится информация о текущем состоянии электронной таблицы. В правой части выводится информация о состоянии управляющих клавиш Caps Lock и NumLock.

Рабочая книга, рабочий лист

Информация в электронной таблице сохраняется в виде *рабочих книг.* Имя книги выводится в строке заголовка. Рабочая книга состоит из листов различного типа. Максимально возможное число листов в рабочей книге - 256.

Рабочий лист состоит из пронумерованных строк и столбцов. Столбцы рабочих листов озаглавлены латинскими буквами от A до Z и их комбинациями, например AA, AB, IU, IV. Строки пронумерованы цифрами. Рабочий лист содержит 256 столбцов и 65536 строк. На пересечении строк и столбцов образованы ячейки. В одной из ячеек расположен *контур выделения* - ***курсор*** электронной таблицы. Рабочий лист имеет номер, который указан на ярлыке (рис. 6.1). Если щелкнуть правой кнопкой мыши по ярлыку, то откроется контекстное меню с перечнем команд для управления рабочим листом. Рабочие листы можно добавлять, удалять, копировать, переименовывать, перемещать, группировать, разгруппировывать.

Если требуется внести одинаковую информацию на несколько листов, то их можно сгруппировать. Для группировки листов нажмите клавишу Shift и щелкните мышью по ярлычкам группируемых листов. Чтобы разгруппировать рабочие листы, выделите их и выберите в контекстном меню команду Разгруппировать.

Ячейка

Основным элементом таблицы является ячейка.

Ячейка - область, образованная пересечением строки и столбца. Она обозначается номером столбца и строки, на пересечении которых находится. Например, A1, IV9999.

Диапазон (группа, блок) - непрерывная область ячеек, обозначенная номерами начальной и конечной ячеек, разделенных двоеточием или точкой, например, A1:C10, D8.H12. Ячейке или диапазону может быть присвоено уникальное имя.

Ячейка характеризуется следующими параметрами: адрес, содержание, значение, формат, статус.

Адрес ячейки может быть абсолютным, относительным и смешанным: относительный адрес: A1, E7; абсолютный адрес: $A$1, $E$7; смешанный адрес: $A1, A$1. Абсолютный адрес ячейки не меняется в операциях копирования, вставки или удаления ячеек, строк и столбцов. Если ячейке присвоен смешанный адрес, то при копировании будет

меняться только тот параметр, перед которым не стоит знак $. Например: $D6 - при копировании ячейки будет меняться только номер строки; D$5 – при копировании будет меняться только адрес столбца.

Ячейке или диапазону ячеек может быть присвоено имя. Присвоение или изменение имени осуществляется командой ***Вставка, Имя***. Для присвоения имени ячейке или диапазону ячеек необходимо:

1. Выделить ячейку (диапазон ячеек);
2. Ввести команду ***Вставка, Имя***, ***Присвоить***;
3. Ввести в строке ввода диалогового окна имя ячейки и щелкнуть **ОК**.

Для удаления имени ячейки введите команду ***Вставка, Имя***, ***Присвоить*** выделите в диалоговом окне имя удаляемой ячейки, щелкните команду ***Удалить*** и ***ОК***.

В формулах возможны ссылки на адрес ячейки или на ее имя. Имя ячейки используется как абсолютный адрес.

Содержание ячейки. Содержание ячейки - это то, что вводится в нее через строку ввода. Поэтому ячейка может быть либо пустая, либо содержать данные: текст, текстовую константу, формулу, дату, время.

Значение ячейки. Значением ячейки могут быть число, текстовая константа, дата, время, сообщения об ошибках. Значением пустой ячейки и ячейки, содержащей текст, является ноль.

Число может быть представлено в виде целого числа (123), вещественного числа с десятичной точкой (0,0001785) или в экспоненциальной форме (1,785E-4). Дробная часть числа отделяется от целой части запятой.

Текстовая константа - это строка символов, используемая в выражениях как операнд, при вводе текстовой константы она заключается в скобки и в кавычки, например, ("ноябрь").

Дата - значение функции дата

Сообщения об ошибках:

#ДЕЛ/0! - деление на ноль;

#ИМЯ? - не определено имя переменной в формуле;

#Н/Д! - нет допустимых значений, аргумент функции не может быть определен;

#ПУСТО! - итога не существует;

#ЧИСЛО! - избыточное число, либо неверное использование числа, например, КОРЕНЬ(-1);

#ССЫЛКА! - неверная ссылка; ячейка, на которую она сделана, в рабочем листе не существует;

#ЗНАЧ! - неправильный тип аргумента; например, использование текста там, где необходимо число.

Если в формуле использовано одно из этих ошибочных значений, результат формулы также будет ошибочным. Ошибочные значения распространяются по всему рабочему листу, помечая все значения, зависящие от них, как некорректные. В этом случае достаточно найти и исправить ошибку, чтобы все остальные ячейки, связанные с ячейкой содержащей ошибку, восстановили свое значение.

Формат ячейки. К формату ячейки относятся ширина, режим отображения формул, формат отображения числовых величин, размещение содержимого ячейки, шрифт, цвет, границы, статус ячейки. Настройка параметров ячейки осуществляется с помощью окна диалога Формат ячеек, которое вызывается командой ***Формат, Ячейка*** (рис. 6.2).

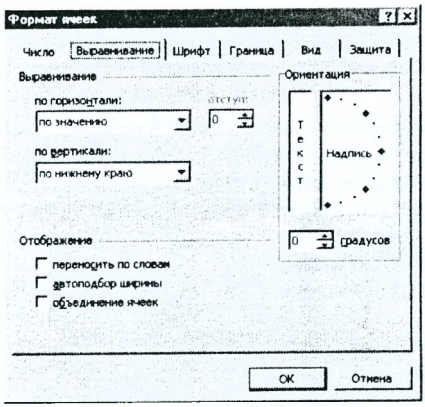


Рис. 6.2. Окно диалога для настройки свойств ячейки

Ширина ячейки может быть от 1 до 255 символов, по умолчанию - 9 символов, а высота – 409 точек. Длина записи для содержимого ячейки – до 32767 символов, в том числе 1024 символа отображаются в ячейке, все символы отображаются в строке формул.

Режим отображения формул: формула или значение. По умолчанию - значение. Для перехода к режиму отображения формул необходимо ввести команду ***Сервис, Параметры***, выбрать закладку ***Вид*** и в группе ***Параметры окна*** установить флажок ***Формулы***.

Формат отображения числовых величин: в виде целого (16, 154) или вещественного числа (1,1755, 5,439), в показательной форме (1,45E-4), в денежном формате (345,32) или ($345,32), в процентном формате (35%). При представлении числа в процентном формате, введенное число делится на сто. Изменение способа представления чисел, даты и времени осуществляется командой ***Формат, Ячейки, Число.***

Размещение содержимого ячейки. Содержимое ячейки может быть размещено справа, слева, по центру. По умолчанию текст прижимается к левому краю, значения - к правому краю. Текст может быть размещен горизонтально, вертикально или под определенным углом. Управление размещением содержимого ячейки осуществляется командой ***Формат, Ячейки, Выравнивание***. Важное значение имеет флажок ***Переносить по словам***. При установке этого флажка текст будет автоматически переноситься в пределах установленной ширины столбца. Если флажок Переносить по словам сброшен, то вводимый текст располагается в одну строку и, если соседняя ячейка заполнена, на экране будет видна только часть текста, умещающаяся в ячейке.

Шрифт. Параметры шрифта: начертание, стиль, цвет, интервал между символами, высота шрифта и другие эффекты, относящиеся к форматированию шрифта, устанавливаются с помощью команды ***Формат, Ячейки, Шрифт***.

Границы ячейки. Стили обрамления и заполнения ячеек устанавливаются командами ***Формат, Ячейки, Границы*** и ***Формат, Ячейки, Вид***, соответственно.

Статус ячейки. Ячейка может иметь два статуса: защищена или не защищена. В защищенную ячейку нельзя внести информацию или изменить ее содержание. Установка режима защиты осуществляется командой ***Формат, Ячейки, Защита***. Режим защиты ячейки вступает в силу только после защиты листа командой ***Сервис, Защита, Защитить лист***. Для отмены защиты ячейки достаточно отменить защиту листа командой ***Сервис, Защита, Снять защиту лист***.

Курсор таблицы

Курсор таблицы или контур выделения представляет собой двойную рамку, окаймляющую всю ячейку (рис. 6.1). В правом нижнем углу рамки на пересечении сторон располагается маленький квадрат - "*маркер заполнения*", используемый при заполнении ячеек рядом данных с постоянным шагом или при копировании. Для перемещения курсора используются клавиши управления перемещением курсора, а также клавиши Home - перейти в первую ячейку строки, Ctrl+Home – перейти в ячейку A1, End+клавиши управления пере-

мещением курсора - последняя занятая ячейка в соответствующем направлении, используются также клавиши прокрутки и ряд других комбинаций клавиш. Непосредственная адресация осуществляется вводом адреса ячейки в поле "Адрес ячейки" Строки Формул.

Ввод данных

Данные вводятся в Строку Формул или непосредственно в ячейку. В первом случае выделите ячейку, в которую вводятся данные, и щелкните по Строке Формул. Введите нужную информацию. Для окончания ввода нажмите клавишу Enter или щелкните кнопку ✓ Строки Формул. Во втором случае выделите ячейку и вводите данные прямо в ячейку. По окончании ввода данных нажмите клавишу Enter.

При редактировании строки ввода используются клавиши:

1. "←", "→" - сдвиг курсора строки ввода на один символ в соответствующем направлении;
2. *Insert* - включение режима вставки символов;
3. *Delete* - удаление символа в позиции курсора;
4. *Home, End, Tab* - переход в начало или конец текста;
5. *SpaceBar* (ПРОБЕЛ) - сдвиг вправо с удалением символов или без удаления символов , в зависимости от режима Вставка/Замена;
6. *BackSpace* (ВОЗВРАТ НА ШАГ) - удаление символа слева от курсора;
7. *Esc* - удаление вводимого текста.

Для очистки ячейки выделите ее и нажмите клавишу Delete или Пробел и Enter. Очистить ячейку можно командой ***Правка, Очистить***. После ввода этой команды откроется дополнительное меню с запросом, что очищать: ***Все, Форматы, Содержимое, Примечания***.

Признаком текста при вводе данных является апостроф ('), например: 'Сводная ведомость, 'Электронная таблица. По умолчанию вводимые данные воспринимаются как текст.

Ввод даты. Дата вводится в формате ДД.ММ.ГГ или ДД.ММ.ГГГГ: день, месяц, год (17.05.99). В качестве разделителя используется точка. Электронная таблица позволяет выводить дату на экран в различных форматах.

Ввод текстовых констант. Для ввода текстовых констант необходимо ввести символ (=) и текст в кавычках. Для преобразования чисел или числовых значений выражений в текстовые константы следует воспользоваться функцией ТЕКСТ().

Тип данных в ячейке определяется при первом вводе. Для изменения формата ячейки используется команда ***Ячейка*** из меню ***Формат***. Например, если в ячейку введена дата, то для ввода числа необходимо изменить формат ячейки командой: ***Формат, Ячейка.*** Затем выбрать закладку ***Число***, выбрать в окне "***Числовые форматы***" тип "***Общий***" или "***Числовой***".

Признаком формулы является знак "=". Если при вводе формулы допущена ошибка, то программа выдает сообщение об ошибке. При вводе формулы без знака равно, программа воспринимает вводимые данные как текст. Адреса ячеек вводятся только латинскими символами.

> ***Совет:*** *чтобы избежать ошибок при записи адресов ячеек выбирайте их мышью. Установите курсор в точку ввода Строки формул и щелкните мышью по ячейке, адрес которой надо вставить в формулу.*

При вводе вещественных чисел, в отличие от других языков програм-мирования высокого уровня, используется десятичная запятая, а не точка. Символ разделитель может быть изменен настройками Windows (Пуск, Настройка, Панель управления, Языки и стандарты).

Для ввода формул или ознакомления с функциями Excel можно использовать Мастер функций. Для этого необходимо щелкнуть инструмент f_x в стандартной панели инструментов или воспользоваться командой ***Вставка, Функция.***

Примеры записи формул:

=A2+2 - сложение; =24-12 - вычитание;

=F35/B7*$A$2 - делит значение ячейки F35 на значение ячейки B7 и умножает на значение ячейки A2. У ячейки A2 указан абсолютный адрес;

=СТАВКА*МЕСЯЦ - перемножаются значения, содержащиеся в ячейках с именами СТАВКА и МЕСЯЦ;

=ЕСЛИ(A2<B2;C3;D2*E17) - условное выражение. Если значение в ячейке A2 меньше значения в ячейке B2, то результат будет равен значению ячейки C3, иначе произведению значений ячеек D2 и E17.

При записи формул, для указания адреса ячеек, значения которых не должны изменяться при копировании формул, обязательно используйте абсолютный адрес.

Для ускорения ввода признака абсолютного адреса - символа $ можно воспользоваться функциональной клавишей **F4**: установите курсор строки ввода в любом месте адреса ячейки и нажмите клавишу F4.

Редактирование содержимого ячейки

Для редактирования содержимого ячейки необходимо выделить ее, при этом содержание ячейки отображается в Строке формул. Щелкните мышкой по Строке формул и вносите необходимые изменения. Для окончания редактирования данных нажмите клавишу Enter или щелкните по кнопке ✓ Строки Формул.

Приемы работы с электронной таблицей

- Выбор ячейки. Установите курсор на ячейку или щелкните мышью.
- Выбор группы ячеек. Установите указатель мыши на первую ячейку группы (области), нажмите левую клавишу и протащите указатель по всем ячейкам группы. В конце области выделения отпустите клавишу мыши. Эту же операцию можно выполнить с помощью клавиш управления перемещением курсора при нажатой клавише Shift.

Для выбора нескольких несвязанных областей необходимо нажать и удерживать клавишу Ctrl, а затем выделить требуемые области.

Для отмены выделения ячеек щелкните мышью по чистому полю в любом месте экрана.

- Выбор строк и столбцов. Для выбора одной строки (столбца) щелкните мышью по номеру строки. Для выбора группы строк (столбцов) установите указатель мыши на номер первой строки, нажмите клавишу мыши и протащите ее указатель по всем строкам выделяемой группы. Отпустите клавишу мыши.
- Изменение размеров строк и столбцов. Для изменения высоты (ширины) строки (столбца) зацепите границу строки, т.е. подведите указатель мыши к границе строки так, чтобы он изменил свою форму на двунаправленную стрелку, нажмите левую клавишу мыши и протащите указатель мыши в нужном направлении до установки требуемых размеров строки. При уменьшении размеров до нуля строки будут спрятаны. Для восстановления спрятанных столбцов, выделите столбцы слева и справа от спрятанных и введите команду ***Формат, Столбец, Отобразить***. Чтобы восстановить спрятанные строки, выделите строки сверху и снизу от спрятанных строк и введите команду ***Формат, Строка, Автоподбор*** высоты или ***Формат***, ***Строка, Отобразить***.
- Вставка, удаление строк, столбцов, ячеек. Для вставки требуемого количества строк (столбцов) выделите столько строк, сколько нужно вставить, и выберите команду ***Вставка***, ***Строка***. Для удаления строк (столбцов) выделите нужные строки и выбери-

те команду ***Удалить*** из меню ***Правка.*** При вставке (удалении) ячеек необходимо дополнительно указать способ освобождения (заполнения) места для ячейки: со сдвигом ячеек вниз (вверх) или вправо (влево).

▪ Форматирование ячеек. Для изменения формата ячейки (группы ячеек) выделите ячейку, выберите команду ***Ячейки*** из меню ***Формат***. Щелкните нужную закладку и установите требуемые параметры. Для окончания работы щелкните кнопку ***ОК***. Форматировать ячейки можно как до, так и после ввода данных. С целью экономии времени применяйте формат сразу к группе ячеек после ввода данных.

Копирование ячеек

При копировании ячеек можно использовать команды ***Копировать*** и ***Вставить*** меню ***Правка***, команды контекстного меню, одноименные кнопки панели инструментов или мышь.

Копирование с использованием маркера автозаполнения

Если копирование осуществляется в соседние ячейки, то его удобно выполнить с использованием маркера автозаполнения:

- выделите копируемую ячейку;
- зацепите мышью за маркер автозаполнения (подведите курсор к черному квадратику в правом нижнем углу курсора таблицы так, чтобы указатель мыши превратился в черный крестик) и протащите указатель мыши по всем ячейкам назначения.

Копирование с помощью команд меню Правка

Для копирования с помощью команд меню Правка или кнопок панели инструментов необходимо выполнить следующее:

- выделить ячейку или блок ячеек, подлежащих копированию;
- выбрать команду Копировать из меню Правка;
- выделить начальную ячейку, куда будут копироваться данные;
- выбрать команду Вставить из меню Правка;
- отменить выделение блока. Чтобы удалить мерцающий контур у копируемых ячеек, щелкните по Строке формул или нажмите клавишу Esc.

Копирование с помощью мыши

Для копирования содержимого ячеек, содержащих текстовую информацию или формулы с абсолютными адресами данных с помощью мыши выделите ячейку или блок ячеек, подлежащих копированию, нажмите и удерживайте клавишу Ctrl, перетащите выделенные ячейки на новое место, отпустите кнопку мыши, а затем отпустите кнопку Ctrl.

Для копирования формул, содержащих ссылки на ячейки с относительными или смешанными адресами, следует воспользоваться командой ***Правка***, ***Специальная вставка***:

- выделите копируемые ячейки;
- укажите место вставки;
- введите команду Правка, Специальная вставка и щелкните по кнопке ***Вставить связь.***

В этом случае в формуле сохраняются ссылки на ячейки содержащие данные. При изменении данных в ячейках автоматически будут меняться значения в ячейках источника данных и в ячейках назначения. При использовании команды Специальная вставка можно выполнять арифметические операции сложения, вычитания, умножения и деления с данными источника данных и данными ячейки назначения, можно копировать только значения или формулы.

Оформление таблицы

Для оформления таблицы: обрамления, заливки цветом, - можно воспользоваться

закладками Граница и Вид окна диалога Формат ячеек или кнопками Границы и Цвет заливки панели инструментов Форматирование, или воспользоваться командой Автоформат меню Формат.

Для оформления таблицы с помощью команды Автоформат выполните следующее:

- выделите таблицу и введите команду ***Формат***, ***Автоформат***;
- выберите нужный стиль оформления и щелкните по кнопке ОК.

Условное форматирование

Часто бывает необходимо выделить какие-то результаты, чтобы обратить на них внимание: минимальные запасы сырья и материалов, предельно допустимые значения параметров среды обитания и др. В этом случае на помощь приходит ***Условное форматирование***.

Выделите ячейки, значения в которых необходимо контролировать, и введите команду Формат, Условное форматирование. Открывается окно диалога Условное форматирование (рис. 6.3). В списке "Условие1" выберите "значение" или "формула". Во втором списке выберите отношение "равно", "не равно", "меньше или равно" и др. В следующем списке укажите значение, формулу, с которой должно сравниваться значение в выделенных ячейках, или ссылку на ячейку. Щелкните по кнопке Формат... . Откроется окно диалога Формат ячеек, которое позволяет настроить параметры шрифта, границы, вид (заливку ячеек цветом), узор. Пример оформления ячеек демонстрируется в окне "Отображение ячейки при выполнении условия". Для завершения работы щелкните по кнопке ОК. Пример выполнения условного форматирования приведен на рис. 6.4.

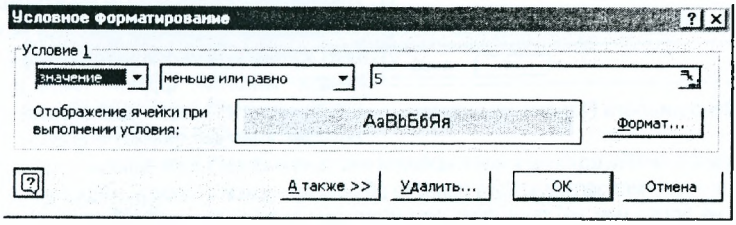


Рис. 6.3. Окно диалога Условное форматирование

Наименование	Наличие запасных частей
Втулка	10
Поршень	15
Карбюратор	5
Тормозные накладки	10
Коробка передач в сборе	2

Рис. 6.4. Пример условного форматирования

Фиксация шапки таблицы

При работе с таблицами, содержащими большое число строк и столбцов, возникают затруднения, так как исчезает либо шапка таблицы, либо левая колонка. Excel имеет средства, позволяющие зафиксировать шапку таблицы, левую колонку или и то и другое вместе. Для фиксации шапки таблицы выделите первую строку ниже той, которую хотите зафиксировать, и введите команду ***Окно, Фиксировать подокна***. Для фиксации левой колонки выделите первую колонку справа от той, которую хотите зафиксировать, и введите команду ***Окно, Фиксировать подокна***. Для одновременной фиксации и шапки таблицы и левой колонки выделите ячейку справа и снизу от фиксируемых строк и колонок и введите команду ***Окно, Фиксировать подокна.*** Для отмены фиксации подокон выберите команду ***Окно, Отменить фиксацию***.

Предварительный просмотр

Перед печатью необходимо убедиться, что данные размещены на листе аккуратно, не выходят за границы листа. Возможно потребуется каким-то образом изменить размещение материала на странице, уплотнить текст, чтобы сократить число страниц. Для этой цели в меню Файл имеется команда Предварительный просмотр, одноименная кнопка имеется и на панели инструментов Стандартная. Для просмотра таблицы введите команду ***Файл, Просмотр.***

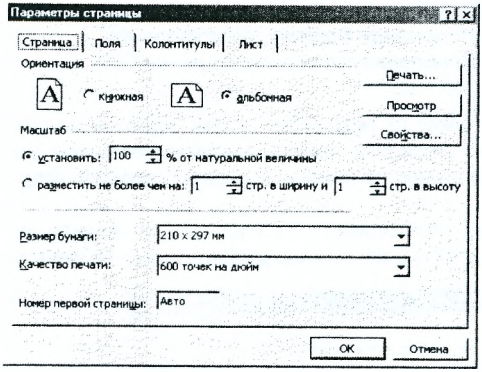


Рис. 6.5. Окно диалога Параметры страницы

Для изменения параметров страницы выберите команду ***Страница*** из меню команды ***Просмотр*** или команду ***Параметры страницы*** из меню ***Файл***.

Закладка ***Страница*** (рис. 6.5) позволяет изменить ориентацию стра-ницы при печати, масштаб, формат листа и др. Страница может иметь два способа ориентации: книжный формат и альбом. При книжном формате печать страницы идет поперек листа, при ориентации альбом - вдоль листа. В обоих случаях лист загружается в принтер одинаково - узкой стороной. Переключатель "Разместить не более чем на" позволяет автоматически выполнить масштабирование документа на указанное число страниц по ширине и высоте.

Закладка ***Поля*** позволяет установить ширину левого и правого поля, высоту верхнего и нижнего полей и высоту поля для колонтитулов, а также центрирование текста документа на странице.

Закладка ***Колонтитулы*** позволяет создать верхний и нижний колонтитулы или выбрать их из списков.

Закладка ***Лист*** (рис. 6.6) позволяет указать диапазон таблицы, выводимый на печать, какие строки и столбцы печатать на каждой странице, управлять выводом на печать номеров строк и столбцов таблицы, сетки, устанавливать порядок обхода страниц при печати. Не выходя из этого режима можно просмотреть как будет выглядеть страница при печати и, наконец, напечатать документ.

После выхода из режима просмотра в таблице пунктирными линиями будут отмечены границы страниц.

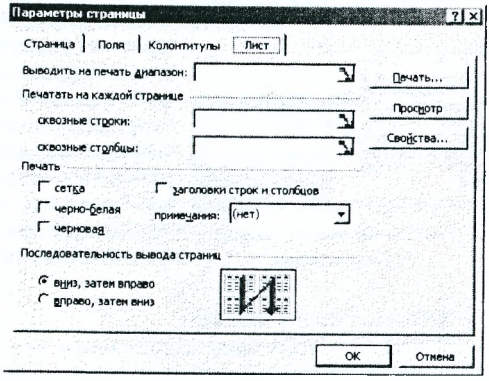


Рис.6.6. Настройка параметров печати листа

Сохранение и печать таблицы

Перед печатью целесообразно сохранить документ на диске, для этого необходимо выполнить следующее: введите команду ***Сохранить как...*** из меню ***Файл***, укажите в строке ввода "Имя Файла" имя файла (расширение имени файла **.xls** оставьте без изме-

нения), выберите, при необходимости, из раскрывающегося списка "Папка" диск и рабочую папку, щелкните по кнопке ОК. Для вывода таблицы на печать выберите команду ***Печать*** из меню ***Файл***, укажите, что печатать: Все, Выделенный диапазон или Номера страниц, Число копий и щелкните кнопку ОК.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как изменить ширину строки (столбца)?
2. Как вставить строку, группу строк (столбцов)?
3. Каким образом устанавливается формат ячейки?
4. Что такое Строка Формул, какие поля она имеет?
5. Где можно увидеть адрес текущей ячейки?
6. Как перейти к новой ячейке?
7. Что такое абсолютный, смешанный и относительный адреса ячейки, как они записываются?
8. Как выполняется копирование ячеек?
10. Как ввести в ячейку текст, формулу?
11. Как сохранить таблицу на диске?
12. Как загрузить таблицу с диска?
13. Как вывести таблицу на печать?
14. Каким образом осуществляется фиксация шапки таблицы?

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с возможностями электронной таблицы и основными приемами работы в ее среде. Единицей учета в Excel является Книга. Книга сохраняется в файле с расширением .xls.

Книга состоит из листов. Основным элементом листа является ячейка. Ввод информации в ячейку осуществляется через строку Формул. Каждая ячейка рассматривается программой как самостоятельный документ, к которому применяются все элементы форматирования.

6.2. РАЗРАБОТКА ЭЛЕКТРОННЫХ ТАБЛИЦ

6.2.1. ТИПЫ ПОЛЕЙ ЭЛЕКТРОННОЙ ТАБЛИЦЫ

Таблица в Excel независимо от ее назначения, имеет четыре поля:

1 - поле описания задачи, состоящее из клеток с текстовой информацией, отражающей наименование и назначение ЭТ; глобальные параметры таблицы; описание строк и столбцов;

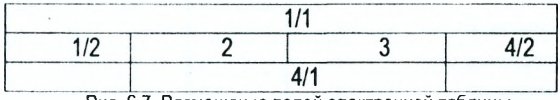


Рис. 6.7. Размещение полей электронной таблицы

2 - поле исходных данных, содержащее клетки с числовой информацией, не изменяющейся в процессе расчета таблицы;

3 - поле расчетных формул, содержащее промежуточные результаты. Операндами в этих клетках являются имена клеток с числовыми данными из полей 1 и 2;

4 - поле формирования результатов расчета ЭТ, которое может содержать как клетки с формулами конечных результатов, так и создаваться процедурой копирования при многовариантном расчете.

Расположение полей зависит от содержания решаемой задачи, объема вычислений, объема исходных данных и способа их ввода и ряда других факторов. При этом необходимо исходить из общего принципа: расположение полей должно обеспечивать наглядность представления материала и доступность данных, удобство их ввода, использования и корректировки.

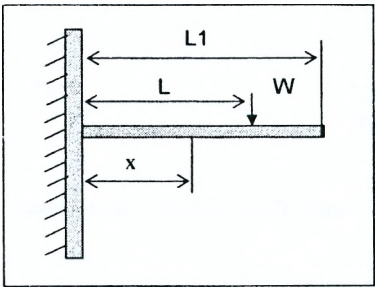


Рис.6.8. К расчету деформации балки

Пример размещения полей приведен на рис. 6.7.

Рассмотрим пример разработки таблицы для расчета деформации балки.

Пример 6.1. Рассчитать деформацию балки закрепленной одним концом на вертикальной опоре (рис. 6.8). Тип балки I – образная, ширина 0,3 м, площадь сечения Z =2 м^2, линейная плотность d - 80 кг/м, момент инерции – I = 3.68 м^4, модуль упругости E=5.3 кг/м^2. Нагрузка приложена на расстоянии L от закрепленного конца, величина нагрузки W. Длина балки L1.

Расчетные формулы:

Напряжение: S=W/Z*(L-x) для x<L (6.1)

Прогиб:

$$y = \frac{Wx^2}{6EI}(3L - x), \text{ если } x<L \text{ и } y = \frac{WL^2}{6EI}(3x - L), \text{ если } x>=L \quad (6.2)$$

Пример разработки таблицы приведен на листинге 6.1.

Листинг 6.1. Разработка таблицы

	A	B	C	D	E	F	G
1	18 Март, 2005						
2	**Расчет деформации балки**						
3	**Исходные данные:**				**Расчет параметров**		
4	Тип балки	Обознач.	I - образная		x	Напряжение	Прогиб
5	Длина, м	L1	3,6		0	91	0
6	Положение нагрузки, м	L	2,6		0,3	80,5	0,403763
7	Нагрузка, кг	W	70		0,6	70	1,550451
8	Ширина, м	B	0,3		0,9	59,5	3,34316
9	Площадь сечения, м^2	Z	2		1,2	49	5,684988
10	Линейная плотность, кг/м^2	d	80		1,5	38,5	8,47903
11	Момент инерции, м^4	I	3,68		1,8	28	11,62838
12	Модуль упругости, кг/м^2	E	5,3		2,1	17,5	15,03615
13					2,4	7	18,60541
14	S=W/Z(L-x), если x<L, иначе 0				2,7	0	22,23988
15	y=Wx^2/(6EI)*(3*L-x), если x<L, иначе				3	0	25,87914
16	y=WL^2/(6EI)*(3*x-L)				3,3	0	29,51839
17					3,6	0	33,15764

Порядок работы

1. Установите курсор в ячейку A1, щелкните по строке ввода – в строке ввода поя-

вится мигающий курсор. Введите дату, например: 18.03.05 и нажмите клавишу Enter или щелкните по кнопке ✓ в Строке формул. Измените формат представления даты командой ***Формат, Ячейка,*** откройте закладку ***Число***, выберите в списке "Числовые форматы" слово Дата, а в списке "Тип" нужный формат представления даты.

2. Введите наименование задачи в ячейку B2. Выделите ячейки A2:F2 и объедините их щелчком по соответствующей кнопке на панели инструментов Форматирование. Введите команду ***Формат, Ячейка***, на закладке Шрифт установите требуемый размер шрифта и начертание. Выравнивание текста по горизонтали выполните с помощью закладки Выравнивание или с помощью соответствующих кнопок на панели инструментов Форматирование.

3. Внесите данные в ячейки A4:C12 согласно Листингу 1. Выполните обрамление блока ячеек A4:C12, используя кнопку Границы панели инструментов Форматирование или закладку Границы окна диалога Формат ячеек.

4. Оформите аналогичным образом шапку таблицы Расчет параметров.

5. Внесите данные в ячейки E5:E17. Для ускорения ввода данных используйте маркер автозаполнения курсора. Введите в ячейку E5 значение 0, а в ячейку E6 – 0,3. Выделите ячейки E5:E6 мышью. Зацепите мышью за маркер автозаполнения и протащите по ячейкам E7:E17.

6. Внесите в ячейку F5 формулу (6.1) со ссылками на адреса ячеек:

=ЕСЛИ(E5<$C$6;$C$7/$C$9*($C$6-E5);0) (6.3)

Ссылки на ячейки C6, C7, C9 записаны в формуле с абсолютным адресом, так как эти данные не должны меняться при копировании.

Функция ЕСЛИ используется для выбора решения из двух альтернатив и имеет следующий синтаксис:

ЕСЛИ(<условие>;<выражение1>;<выражение2>)

Функция работает следующим образом: программа проверяет условие и, если оно выполняется (истинно), то возвращает результат согласно Выражению1, в ином случае возвращается результат согласно Выражению2.

Например. Требуется проверить равенство значений в ячейках A1 и B1 и вывести результат в ячейку C1.

Запишем в ячейку C1 формулу:

ЕСЛИ(A1=B1;"Выражение истинно";"Выражение ложно")

В зависимости от значений в ячейках A1 и B1 получим следующие результаты:

Листинг 6.2. Использование функции ЕСЛИ

	A	B	C	D
1	5	5	Выражение истинно	

	A	B	C	D
1	5	7	Выражение ложно	

7. Внесите в ячейку G5 формулу (6.2) со ссылками на адреса ячеек:

=ЕСЛИ(E5<$C$6;$C$7*E5^2/(6*$C$12*$C$11)*(3*$C$6-E5);$C$7*$C$6^2/(6*$C$12*$C$11)*(3*E5-$C$6)) (6.4)

8. Скопируйте формулы из ячеек F5, G5 в ячейки F6:G17, используя механизм автозаполнения.

6.2.2. ФУНКЦИИ ЭЛЕКТРОННОЙ ТАБЛИЦЫ

Excel имеет 11 категорий различных функций: математические/ тригонометрические; инженерные; логические; текстовые; статистические; функции категории дата/время; функции для работы с базами данных/списками; финансовые; информационные и функции категории ссылки/массивы; функции проверки свойств и значений. Кроме того, Excel содержит большое число надстроечных функций, которые используются для создания компьютерных программ в Excel, а также имеется возможность создания пользовательских функций и программ на Visual Basic for Applications. Можно написать программы на других языках программирования высокого уровня, например, C, FORTRAN и потом вызвать их в Excel.

Вызов функций осуществляется с помощью кнопки панели инструментов f_x или команды ***Функция*** меню ***Вставка***. Эта команда вызывает на экран окно диалога ***Мастер функций*** (рис. 6.9), который обеспечивает выбор функции из списка и пошаговый ввод сложных функций в режиме диалога.

Окно диалога имеет два списка. В левом списке приведены категории функций, а в правом – функции. В списке категорий последней в списке будет категория "Пользовательские функции". В эту категорию попадают функции, созданные пользователем с помощью встроенного языка программирования Visual Basic for Application.

Ниже окон списков выводятся текстовые строки, в которых отображается синтаксис выделенной функции и ее назначение. Слева в нижней части окна диалога расположена кнопка, которая выводит контекстную подсказку по выбранной функции. При щелчке по кнопке ОК на экран выводится окно диалога для ввода данных в шаблон функции.

Для поиска нужной функции выберите в списке "Категории" категорию функции, а в правом – соответствующую функцию. Если неизвестно, к какой категории относится функция, то выберите "Полный алфавитный перечень" и найдите в нем требуемую функцию. Выбранная функция попадает в список "10 недавно использовавшихся". Поэтому при последующем обращении к выбранной функции, ее можно будет найти в этом списке.

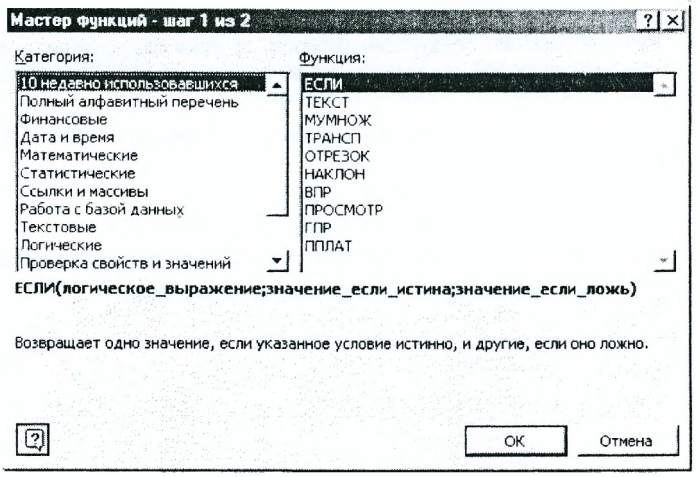


Рис. 6.9. Окно диалога Мастера функций

Некоторые функции приведены в табл.6.1.

С другими функциями можно познакомиться по технической документации или по справочной системе Excel.

В качестве примера использования мастера функций рассмотрим порядок ввода функции ЕСЛИ согласно выражений (6.2), (6.4):

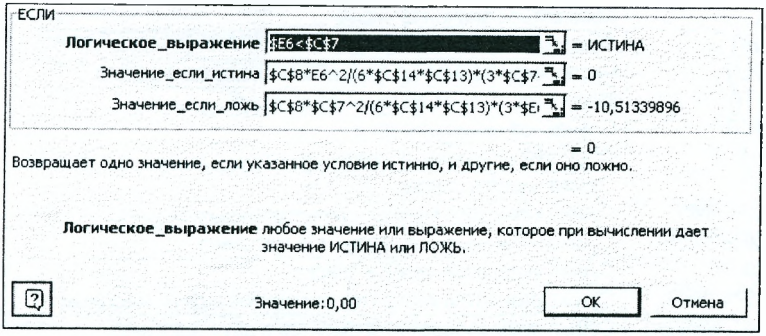


Рис. 6.10. Использование мастера функций

• Выделите ячейку G6, в которую надо поместить выражение. Введите команду Вставка функция или щелкните по одноименной кнопке f_x на панели инструментов стандартная - откроется окно диалога Мастер функций (рис. 6.9).

• Найдите в правом списке функцию ЕСЛИ, выделите ее и щелкните по кнопке ОК – откроется окно диалога для ввода формул (рис. 6.10).

• В окне диалога имеется три строки ввода, в соответствии с синтаксисом функции. При выделении любой строки в нижней части таблицы выводится подсказка о назначении данной строки. Первая строка служит для ввода логического условия, вторая – для ввода выражения соответствующего истине, и третья строка – для ввода выражения, соответствующего отрицательному результату сравнения (ложь).

• Введите в первую строку логическое условие E6<C7. Введите во вторую строку выражение "истина":

$C$8*E6^2/(6*$C$14*$C$13)*(3*$C$7- E6)

Введите в третью строку выражение "ложь"

$C$8*$C$7^2/(6*$C$14 *$C$13)*(3*E6-$C$7)

Если формулы введены правильно, то сразу же можно увидеть результат. Так как условие истинно, то функция возвращает результат согласно первому выражению (строка 2).

• Для завершения работы щелкните по кнопке ОК.

При достаточном навыке формулы можно вводить и без использования мастера функций. Достоинством использования мастера функций является то, что всегда можно получить оперативно подсказку по каждому полю ввода.

6.2.3. ГЕНЕРИРОВАНИЕ ДАННЫХ

Часто бывает необходимо сгенерировать последовательность чисел, дат.

Для этой цели можно использовать механизм автозаполнения. Чтобы заполнить несколько ячеек прогрессией, необходимо записать в смежные ячейки данные, отличающиеся на величину шага, выделить эти ячейки, и перетащить маркер заполнения выделенного диапазона ячеек. Можно также воспользоваться командой ***Прогрессия*** программы Excel. Внесите в ячейку начальное значение ряда чисел; выделите область для заполнения, выберите пункт ***Заполнить*** в меню ***Правка***, а затем щелкните пункт ***Прогрессия***.

Таблица 6.1 Функции электронной таблицы

	Математические
ABS()	Абсолютное значение числа
ФАКТР()	Факториал числа
ЦЕЛОЕ()	Число, округленное до ближайшего меньшего целого
ОСТАТ()	Модуль(остаток от деления двух чисел)
СЛЧИС()	Случайное число от 0 до 1
КОРЕНЬ()	Квадратный корень из числа
СУММА()	Сумма чисел в списке
СУММЕСЛИ()	Сумма значений в ячейках, соответствующих заданному критерию
СУММПРОИЗВ()	Сумма произведений элементов массивов
СУММКВ()	Сумма квадратов чисел в списке
СУММРАЗНКВ()	Сумма разностей квадратов элементов в двух массивах
СУММСУММКВ()	Сумма сумм квадратов элементов в двух массивах
СУММКВРАЗН()	Сумма квадратов разностей значений в двух массивах
	Логарифмические функции
EXP()	Число e, возведенное в степень
LN()	Натуральный логарифм числа (основание "e")
LOG()	Логарифм числа по заданному основанию LOG(число,основание)
LOG10()	Логарифм числа по основанию 10
	Тригонометрические функции
ПИ()	Возвращает значение числа π
COS()	Косинус числа
SIN()	Синус числа
TAN()	Тангенс числа
	Обратные тригонометрические функции
ACOS()	Арккосинус числа
ASIN()	Арксинус числа
ATAN()	Арктангенс числа от -π/2 до π/2
ATAN2()	Арктангенс отношения двух чисел (от -π до π)
	Функции преобразования угла
ГРАДУСЫ()	Показатель величины угла в градусах
РАДИАНЫ()	Показатель величины угла в радианах
	Матричные функции
МОПРЕД()	Определитель матрицы
МОБР()	Матрица, обратная заданной
МУМНОЖ()	Произведение двух матриц
ТРАНСП()	Транспонирование матрицы

На экран выводится диалоговое окно Прогрессия (рис. 6.11). Выберите *Тип* прогрессии, *Расположение* в соответствии с выделенной областью, *Шаг* и щелкните по кнопке ОК.

Если известно конечное значение ряда чисел, то введите его в последнюю ячейку выделенного диапазона, тогда шаг прогрессии определится автоматически.

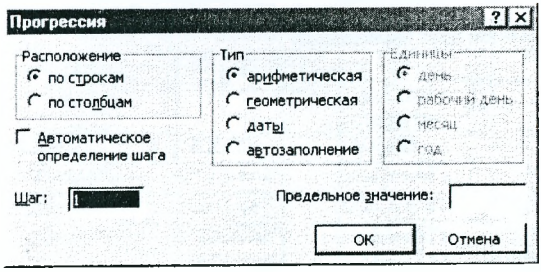


Рис. 6.11. Окно диалога Прогрессия

При выборе типа Даты активизируется группа "Единицы". Тогда можно вывести дни по порядку, рабочие дни в текущем месяце, число дней помесячно или число дней по годам.

Для заполнения ячеек часто используемыми текстовыми записями можно создавать пользовательские прогрессии. Для этого выполните следующее: выберите команду ***Параметры*** в меню ***Сервис*** и откройте затем закладку ***Списки***. В окне списки выберите строку *"Новый список"* и введите данные в поле *"Элементы списка"*. После ввода каждой новой записи нажимайте клавишу Enter. Для завершения работы щелкните по кнопке *Добавить*.

Ряды чисел часто применяются для табулирования функций переменных. В этих случаях *целесообразнее* создать собственную программу генерирования ряда чисел с настраиваемым шагом (Листинг 6.3). Для этого выполните следующее:

- введите в ячейку A1 текст "Начальное значение", а в ячейку B1 начальное значение ряда;
- введите в ячейку A2 текст "Шаг табуляции", а в ячейку B2 значение шага табуляции (приращение аргумента);
- запишите в ячейку A4 начальное значение ряда путем ссылки на ячейку B1: выделите ячейку B4 и запишите в нее формулу: = B1;
- запишите в ячейку A5 формулу арифметической прогрессии A4+$B$2 ;
- определите диапазон ячеек, куда необходимо скопировать формулу (номер начальной и конечной ячеек);
- скопируйте формулу из ячейки A5 в остальные ячейки диапазона.

Листинг 6.3.
Табулирование функции

	A	B
1	Начальное знач.	1
2	Шаг табуляции	0,5
3	Аргумент	Функция
4	=B1	=SIN(A4)
5	=A4+$B$2	=SIN(A5)
6	=A5+$B$2	=SIN(A6)

6.2.4. ТАБУЛИРОВАНИЕ ФУНКЦИЙ

Табулирование функций с использованием операций копирования

Под табулированием понимают конструирование, вычисление и составление различных математических таблиц (см. раздел 3.3.3).

Пример табулирования функции одной переменной приведен на Листинге 6.3. Для выполнения операции табулирования необходимо:

- сгенерировать ряд значений аргумента на заданном интервале;
- записать в соседний столбец справа расчетную формулу зависимости функции от аргумента;
- скопировать расчетную формулу во все ячейки требуемого диапазона изменения аргумента.

Пример табулирование функции двух переменных $z=2x+y^2$ приведен на Листинге 6.4.

Листинг 6.4. Табулирование функции двух переменных

	A	B	C	D	E
1	Шаг 1-го аргумента	0,5	Шаг 2-го аргумента	0,2	
2		0,1	B2+$D$1	C2+$D$1	D2+$D$1
3	1	2*$A3+B$2^2	2*$A3+C$2^2	2*$A3+D$2^2	2*$A3+E$2^2
4	A3+$B$1	2*$A4+B$2^2	2*$A4+C$2^2	2*$A4+D$2^2	2*$A4+E$2^2
5	A4+$B$1	2*$A5+B$2^2	2*$A5+C$2^2	2*$A5+D$2^2	2*$A5+E$2^2
6	A5+$B$1	2*$A6+B$2^2	2*$A6+C$2^2	2*$A6+D$2^2	2*$A6+E$2^2
7	A6+$B$1	2*$A7+B$2^2	2*$A7+C$2^2	2*$A7+D$2^2	2*$A7+E$2^2
8	A7+$B$1	2*$A8+B$2^2	2*$A8+C$2^2	2*$A8+D$2^2	2*$A8+E$2^2
9	A8 +$B$1	2*$A9+B$2^2	2*$A9+C$2^2	2*$A9+D$2^2	2*$A9+E$2^2

Порядок выполнения операции следующий:

- запишите в ячейку A3 начальное значение аргумента X;
- запишите в ячейку B2 начальное значение аргумента Y;
- запишите в ячейки A4 и C2 формулы для генерирования рядов значений аргументов;
- скопируйте в ячейки A5:A9 формулу для вычисления аргумента X из ячейки A4;
- скопируйте в ячейки D2:E2 формулу для вычисления Y из ячейки C2;
- запишите в ячейку B3 таблицы расчетную формулу с использованием смешанных адресов ячеек: у первого аргумента зафиксируйте столбец, а у второго аргумента - строку;
- скопируйте формулу во все ячейки блока.

> *Общее правило при копировании формул со смешанными адресами:* если данные находятся в строке, то фиксируется номер строки, а если данные находятся в столбце, то фиксируется номер столбца.

Табулирование функции с использованием модуля

Таблица подстановки

Для табулирования функций одной и двух переменных можно использовать средства Excel: команду ***Таблица подстановки*** из меню ***Данные***. Однако, с точки зрения автора, алгоритмы табулирования функции с использованием этой команды не отвечают требованию массовости и не дают выигрыша во времени. Тем не менее, рассмотрим алгоритм использования модуля Таблица подстановки.

Пример 6.2. Протабулировать функцию SIN(x) на интервале от –ПИ/2 до ПИ/2 с шагом 0,5.

Решение:

- введите в ячейки A1, A3, A5, C1 (Листинг 6.5) текст Ячейка ввода, Начальное значение, Шаг табуляции, Ячейка ввода формулы;

- введите в ячейку ввода A2 произвольное число, например 0, (это число не влияет на результат табулирования);

Листинг 6.5. Использование команды Таблица подстановки

	A	B	C
1	Ячейка ввода		Ячейка ввода формулы
2	0		=SIN(A2)
3	Начальное значение	-1,5708	-1
4	-1,5708	-1,0708	-0,87758
5	Шаг табуляции	-0,5708	-0,5403
6	0,5	-0,0708	-0,07074
7		0,429204	0,416147
8		0,929204	0,801144
9		1,429204	0,989992
10		1,929204	0,936457

- ведите в ячейку A4 начальное значение аргумента x - -ПИ/2. Для ввода этого числа используем функцию ПИ();

- введите в ячейку A6 значение шага - 0,5;

- введите в ячейку C2 формулу SIN(A2). В качестве аргумента указывается адрес ячейки ввода (A2);

- сгенерируйте в столбце B, начиная с ячейки B3, ряд значений аргумента;

- выделите область B2:C10 и введите команду ***Данные, Таблица подстановки***;

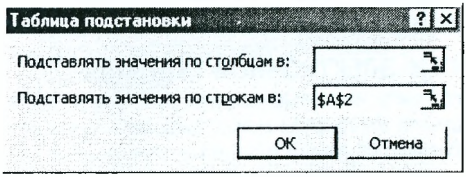


Рис.6.11. Окно диалога Таблица подстановки

- в диалоговом окне Таблица подстановки (рис. 6.11) введите в окно ввода *"Подставлять значения по строкам в..."* номер Ячейки ввода A2 и щелкните кнопку ОК. (Для ввода номера ячейки достаточно активизировать окно ввода щелчком мыши и щелкнуть по ячейке A2). Работа завершена.

Пример 6.3. Протабулировать функцию $2x+y^2$ при x, изменяющимся от 0 до 1 с шагом 0,2, и y изменяющимся от 1 до 4 с шагом 1.

Решение:

- обозначьте ячейку A2 как ячейку ввода по строкам, а ячейку A4 как ячейку ввода по столбцамю Для этого внесите соответствующие записи в ячейки A1 и A3 (см. листинг 6.6);

Листинг 6.6. Табулирование функции двух переменных

	A	B	C	D	E	F
	По строкам	2*x+y^2	1	2	3	4
	1	0	1	4	9	16
	По столбцам	0,2	1,4	4,4	9,4	16,4
	1	0,4	1,8	4,8	9,8	16,8
		0,6	2,2	5,2	10,2	17,2
		0,8	2,6	5,6	10,6	17,6
		1	3	6	11	18

- внесите в столбец B, начиная с ячейки B2, значения аргумента x;

- внесите в строку 1, начиная с ячейки C1 значения аргумента y;

- внесите в ячейку B1 (ячейка на пересечении первого столбца и первой строки будущей таблицы) формулу 2*x+y^2 или, с учетом ссылок на номера ячеек, 2*A2+A4^2;

- выделите область B1:F7 и введите команду Данные, Таблица подстановки;

- внесите в строку ввода *Подставлять значения по столбцам в ...:* номер ячейки A4, а в строку ввода *Подставлять значения по строкам в ...:* номер ячейки A2 и щелкните кнопку ОК. Работа завершена.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каким образом можно сгенерировать ряд чисел, используя маркер заполнения курсора таблицы?
2. Как сгенерировать ряд чисел с арифметической или геометрической прогрессией?
3. Как протабулировать функцию одной переменной?
4. Как протабулировать функцию двух переменных?
5. Опишите алгоритм табулирования функции одной переменной с использованием модуля Таблица подстановки.
6. Опишите алгоритм табулирования функции двух переменных с использованием модуля Таблица подстановки.

ЗАКЛЮЧЕНИЕ

Электронная таблица позволяет генерировать ряды данных: чисел, дат, времени и других последовательностей заданных пользователем с использованием маркера автозаполнения.

Во многих задачах имеется необходимость генерирования значений аргумента и соответствующих значений функции. Эта операция может выполняться как с использованием операции копирования, так и команды Таблица подстановки из меню Данные.

6.3. ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ ЭЛЕКТРОННОЙ ТАБЛИЦЫ

Диаграммы и графики позволяют представить числовые данные, результаты обработки таблиц в наглядной форме. При создании диаграммы можно выделить три этапа: создание таблицы, описание графика на бумаге, описание графика в электронной таблице и его использование.

Хорошо разработанная таблица содержит все элементы, необходимые для описания графика. При описании графика на бумаге необходимо установить соответствие между элементами таблицы и графиком (при достаточном навыке этот этап не обязателен). Если на таблице нет каких-то элементов, то их необходимо описать вне пределов таблицы.

График включает, обычно, следующие элементы: заголовок, обозначение осей, разметку по осям, описание меток (легенда), числовые данные на графике. Кроме того, необходимо определить тип диаграммы, наиболее подходящий для имеющихся данных.

Для построения графиков и диаграмм в электронной таблице используется Мастер диаграмм, который за четыре шага позволяет описать все элементы графика. Имеется возможность редактировать график после построения.

Первое диалоговое окно позволяет выбрать тип диаграммы и ее вид. Мастер диаграмм позволяет использовать 14 стандартных и 20 нестандартных типов диаграмм. Для построения графиков функций необходимо использовать "Точечную" диаграмму. Тип "График" целесообразно использовать для построения линейных диаграмм. Это связано с тем, что график строится отрезками прямых, поэтому при большом шаге табулирования функции будет наблюдаться ступенчатость графика. Выбранный вид диаграммы отображается на экране.

Второе диалоговое окно позволяет ввести диапазон значений для построения диаграммы, а также установить порядок чтения данных: по строкам или столбцам (рис. 6.12). Если диалоговое окно закрывает часть экрана, нужную для выделения области, щелкните кнопку в конце строки ввода диапазона. Окно диалога убирается с экрана, кроме самой строки ввода. Для восстановления диалогового окна после выделения требуемой области снова щелкните кнопку в конце строки ввода диапазона. Вкладка Ряд этого диалогового окна позволяет добавлять данные на диаграмму или удалять данные (рис.6.13).

На этой вкладке имеется окно "Ряд", окно просмотра графика и две или три строки ввода (в зависимости от типа графика). Окно "Ряд" служит для отображения списка графиков. Оно позволяет также добавлять или удалять графики с диаграммы. В строке Имя вводится название графика или адрес ячейки, в которой он находится. В строках Значения X, Значения Y указывается диапазон ячеек, в которых содержатся соответствующие данные.

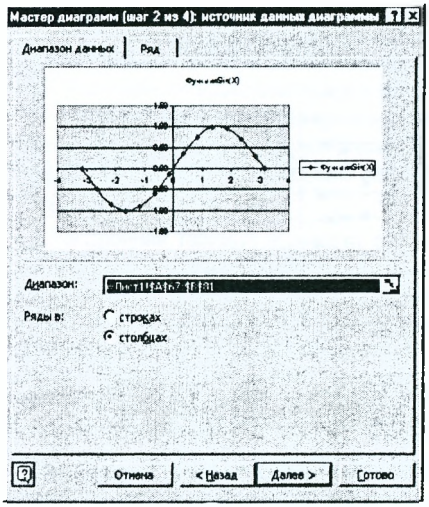


Рис.6.12. Мастер диаграмм, шаг 2

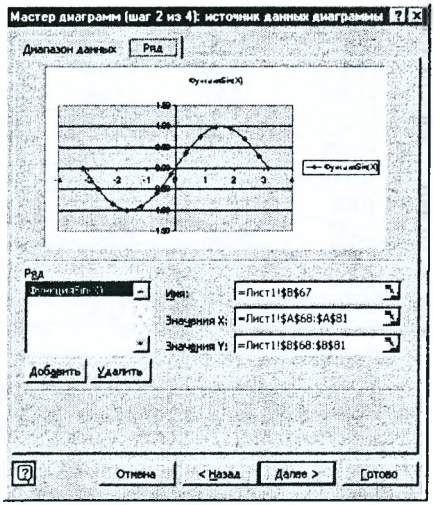


Рис.6.13. Мастер диаграмм, шаг 2

Третье диалоговое окно позволяет описать диаграмму. Оно содержит пять вкладок и позволяет вводить Заголовок диаграммы и заголовки осей, указывать типы осей, выводить на диаграмму сетку, легенду и подписи данных.

Четвертое диалоговое окно позволяет указать место для вывода диаграммы: на текущем рабочем листе или на отдельном листе. В последнем случае предоставляется возможность представить диаграмму в большем формате.

Диаграмму можно перемещать по рабочему листу, изменять ее размеры, копировать. Если щелкнуть дважды мышью по какому-либо элементу диаграммы, то открывается окно диалога для настройки соответствующего элемента диаграммы (цвет линий, стиль линий, тип диаграммы, шрифт и так далее). При щелчке правой кнопкой мыши по элементу диаграммы вызывается контекстное меню для настройки соответствующего элемента.

Пример 6.4. Построение графика функции одной переменной.

Построить графики функций Sin(x) и Cos(x) на отрезке от -ПИ до ПИ. Отрезок разделить на десять равных частей. Пример построения графика приведен на Листинге 6.7.

Листинг 6.7. График функции

	A	B	C	D	E	F	G	H
1	**Построение графика функции**							
2								
3	Интервал:	-3,1416	3,1416					
4	Число точек	10	Шаг табуляции	0,628319				
5	Аргумент	Sin(x)	Cos(x)					
6	-3,142	0,000	-1,000					
7	-2,513	-0,588	-0,809					
8	-1,885	-0,951	-0,309					
9	-1,257	-0,951	0,309					
10	-0,628	-0,588	0,809					
11	0,000	0,000	1,000					
12	0,628	0,588	0,809					
13	1,257	0,951	0,309					
14	1,885	0,951	-0,309					
15	2,513	0,588	-0,809					
16	3,142	0,000	-1,000					

Порядок работы

1. Протабулируйте функции на заданном отрезке (см. пример 6.1.). Шаг табуляции определить как отношение длины отрезка табулирования функции к числу отрезков N: (ПИ() – (-ПИ))/N.

2. Выделите область исходных данных для построения графика функции Sin(x), включая заголовки: A5:B16. Щелкните в стандартной панели инструментов кнопку Мастер диаграмм .

3. Выполните первый шаг: выберите тип диаграммы - точечная и вид диаграммы - плавная кривая с метками точек на графике функции. Щелкните кнопку Далее.

4. Выполните второй шаг: так как диапазон данных был указан заранее, то просто щелкните кнопку Далее.

5. Выполните третий шаг: используя закладки диалогового окна, введите название диаграммы "График функции Sin(x)", название оси X - "X", название оси Y - "Y". Установите, при необходимости, линии сетки основные и промежуточные, определите место расположения легенды, режим вывода на график числовых значений или категорий. Щелкните кнопку Далее.

6. Выполните четвертый шаг: укажите место размещения диаграммы (на отдельном листе или в текущем рабочем листе).

7. Добавьте к графику функции Sin(x) график функции Cos(x). Для этого выполните следующее:

- щелкните правой кнопкой мыши линию графика функции Sin(x) и выберите в контекстном меню команду ***Исходные данные***;
- в окне диалога Исходные данные на закладке Ряд щелкните кнопку Добавить;
- в строку ввода Имя введите щелчком мыши содержание ячейки C5;
- в строку ввода Значения введите диапазон значений функции C6:C16;
- в строку ввода Подписи по оси X введите диапазон значений аргумента A6:A16.
- щелкните кнопку Ок.

Примечание. Можно строить одновременно несколько графиков функций. В рассмотренном примере графики функций Sin(x) и Cos(x) строятся по очереди в учебных целях.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные элементы диаграммы.
2. Опишите порядок построения графиков функций с использованием мастера диаграмм.
3. Как добавить график функции на диаграмму?
4. Как изменить заголовок диаграммы или ее осей?
5. Как изменить стиль линий сетки?
6. Как поместить диаграмму в документ Word?
7. Каким образом перемещается диаграмма по рабочему листу?
8. Как изменить размеры диаграммы?
9. Какой тип диаграммы больше подходит для построения графиков функций?

ЗАКЛЮЧЕНИЕ

Электронная таблица позволяет отображать данные в виде графиков и диаграмм. Для построения диаграмм используется Мастер диаграмм, который предлагает 14 видов стандартных и 17 видов нестандартных диаграмм. Для построения графиков функций целесообразно использовать точечную диаграмму. Мастер диаграмм позволяет производить различные настройки: оформление заголовков, сетки, таблицы данных, подписи данных, легенды. Для автоматического создания легенды рекомендуется перед построением диаграммы или графика выделить таблицу данных вместе с шапками таблицы (заголовками строк и столбцов). Электронная таблица позволяет редактировать графики и диаграммы и их отдельные элементы.

6.4. РАБОТА С МАТРИЦАМИ

6.4.1. ОПЕРАЦИИ С МАТРИЦАМИ

Эектронная таблица позволяет выполнять линейные преобразования матриц: умножение, деление матриц на число, прибавление или вычитание чисел, а также операции над матрицами: сложение, умножение матриц, транспонирование, вычисление определителей. Средствами Excel можно решать и системы линейных алгебраических уравнений. Для этой цели электронная таблица имеет ряд функций для работы с матрицами:

МОБР(массив) – вычисление обратной матрицы;

МОПРЕД(массив) – вычисление определителя матрицы;

МУМНОЖ(массив; массив) – умножение матриц;

ТРАНСП(массив) – транспонирование матриц.

Примеры операций с матрицами приведены на Листинге 6.8. Обратите внимание на разные результаты, получаемые при умножении матриц с использованием оператора умножения "*", и с использованием функции МУМНОЖ. В первом случае каждый элемент матрицы результата равен произведению соответствующих элементов сомножителей, во втором случае каждый элемент матрицы вычисляется по формуле $C(i,k) = \sum_{j=1}^{m} A(i,j) * B(j,k)$, где m – число столбцов в матрице A, матрица C имеет размерность PxQ, P – число строк в матрице A, Q – число столбцов в матрице B.

Алгоритм выполнения операций над матрицами сводится к следующим операциям:

- выделить ячейку или область, если результатом выполнения операции будет матрица, куда будет помещаться результат вычисления (B8:C9);
- ввести в строку ввода символ "=";
- ввести в строку ввода первый операнд, например, область матрицы B (B5:C6);
- ввести в строку ввода символ операции, например, оператор сложения "+";
- ввести в строку ввода второй операнд, например, адрес числа a - B3. Для первого примера на Листинге 6.8 получим выражение {=B5:C6+B3};
- нажать комбинацию клавиш Ctrl+Shift+Enter для вставки формулы в выделенную область.

Листинг 6.8. Примеры операций с матрицами

	A	B	C	D	T	F	G	H	I	J	K	L
1	Прибавление числа к матрице				Умножение матрицы на число				Сложение матриц			
2												
3	a=	2,543			a1=	7,345			C=B+B1	3	6	
4										8	11	
5	B=	2	4		B1=	1	2					
6		5	7			3	4		Умножение матриц			
7												
8	B+a=	4,543	6,543		B1*a1	7,345	14,69		D=B*B1	2	8	
9		7,543	9,543		=	22,035	29,38			15	28	
10												
11	Транспонирование матриц						Использование функции МУМНОЖ					
12												
13	ТРАНСП(B5:C6)			2	5		МУМНОЖ(B;B1)=			14	20	
14				4	7					26	38	

6.4.2. РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Для решения систем линейных алгебраических уравнений применяют аналитические и численные методы.

Электронная таблица Excel не имеет функций для решения систем уравнений, формулы для вычисления матриц необходимо формировать самостоятельно, используя известные методы, например, метод Крамера или метод Гаусса (метод исключения переменных). Однако, используя встроенные функции МОБР, МУМНОЖ и МОПРЕД, эти операции выполняются достаточно легко. Например, можно воспользоваться формулой вычисления вектора неизвестных через обратную матрицу и вектор свободнх членов: $\overline{X} = A^{-1} \bullet \overline{B}$.

Пример 6.4. Решить систему линейных алгебраических уравнений матричным методом (Листинг 6.9)

$$\begin{cases} 65{,}18x + 36{,}31y + 23{,}76z = 86{,}46 \\ -17{,}98x + 23{,}89y + 27{,}54z = -38{,}07 \\ 23{,}75x + 13{,}94y + 48{,}12z = 53{,}97 \end{cases} \qquad (6.5)$$

Решение.

1. Внесите в ячейки B6 – D8 значения коэффициентов при неизвестных.

2. Внесите в ячейки F6 – F8 значения свободных членов системы уравнений.

3. Выделите диапазон ячеек B12: D14 и введите формулу МОБР(B6:D8), для завершения операции ввода нажмите комбинацию клавиш Ctrl+Shift+Enter.

4. Выделите диапазон ячеек F12:F14 и введите формулу МУМНОЖ(B12:D14;F6:F8). Для завершения ввода формулы нажмите комбинацию клавиш Ctrl+Shift+Enter. В ячейках F12 – F14 появятся значения корней уравнения.

ЛИСТИНГ 6.9. РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МАТРИЧНЫМ СПОСОБОМ $X=A^{-1}B$

	A	B	C	D	E	F	G
5		**Матрица Коэффициентов**				***Вектор свободных членов***	
6		65,18	36,31	23,76		86,46	
7		-17,98	23,89	27,54		-38,07	
8		23,75	13,94	48,12		53,97	
9							
10			***Обратная матрица***			***Результат***	
11							
12		0,009	-0,017	0,005		1,67504	
13		0,018	0,030	-0,026		-1,01011	
14		-0,010	-0,001	0,026		0,58746	

Пример 6.5. Решите систему линейных алгебраических уравнений (6.5) методом Крамера (Листинг 6.10).

Решение.

1. Внесите в таблицу расширенную матрицу, то есть запишите в ячейки A2:D4 электронной таблицы коэффициенты при неизвестных и свободные члены;

2. Запишите в A6:C8 коэффициенты матрицы, используя в качестве исходных данных адреса ячеек из расширенной матрицы. Этот метод предпочтительнее простого копирования, так как в этом случае при изменении данных в ячейках расширенной матрицы автоматически изменяются и значения в ячейках дополнительного определителя;

3. Скопируйте два раза коэффициенты матрицы из ячеек A6:C8 в ячейки A10:C12 и A14:C16;

4. Сформируйте из копий матриц дополнительные определители путем замены коэффициентов при неизвестных на вектор свободных членов. При этом также как и в пункте 2 следует ссылаться на адреса ячеек D2 , D3, D4;

5. Запишите напротив первой строки расширенной матрицы в ячейку E2 расчетную формулу для вычисления главного определителя: МОПРЕД(A2:C4);

6. Скопируйте расчетную формулу из ячейки E2 в ячейки E6, E10, E14.

7. Запишите формулы для вычисления неизвестных как отношение соответствующих дополнительных определителей к главному определителю в ячейки G6, G10 и G14.

Листинг 6.10. Решение систем линейных алгебраических уравнений методом Крамера

	A	B	C	D	E	F	G
1	Расширенная матрица				Гл. опред		
2	65,18	36,31	23,76	86,46	85635		
3	-17,98	23,89	27,54	-38,07			
4	23,75	13,94	48,12	53,97			
5	Первый дополнительный определитель						***Результат***
6	86,46	36,31	23,76		143443	X=	1,67504
7	-38,07	23,89	27,54				
8	53,97	13,94	48,12				
9	Второй дополнительный определитель						
10	65,18	86,46	23,76		-86501	Y=	-1,01011
11	-17,98	-38,07	27,54				
12	23,75	53,97	48,12				
13	Третий дополнительный определитель						
14	65,18	36,31	86,46		50308	Z=	0,58746
15	-17,98	23,89	-38,07				
16	23,75	13,94	53,97				

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое расширенная матрица коэффициентов системы линейных уравнений?
2. Что является решением системы линейных алгебраических уравнений?
3. Какие методы применяются для решения СЛАУ в электронной таблице?
4. Напишите алгоритм решения САЛУ методом Крамера.
5. Напишите алгоритм решения САЛУ матричным методом с использованием обратной матрицы.
6. Назовите функции электронной таблицы для работы с матрицами.

ЗАКЛЮЧЕНИЕ

В настоящем разделе мы познакомились с возможностями электронной таблицы по работе с векторами и матрицами. Excel имеет несколько функций для работы с массивами: МОБР, МОПРЕД, МУМНОЖ, ТРАНСП. Эти функции позволяют выполнять элементарные преобразования матриц, а также решать системы линейных алгебраических уравнений известными методами.

6.5. ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОГО АНАЛИЗА

6.5.1. ВЫЧИСЛЕНИЕ ПРОИЗВОДНЫХ ЧИСЛЕННЫМИ МЕТОДАМИ

Если функция задана в виде таблицы, то производные от функции в любой точке могут быть вычислены численными методами по известным приближенным разностным формулам (см. раздел 3.3.1, пример 3.5.) :

$$\frac{dy}{dx} = \frac{y(x+h) - y(x-h)}{2h} \qquad O(h) \qquad (6.6)$$

$$\frac{d^2y}{dx^2} = \frac{y(x+h) - 2y(x) + y(x-h)}{h^2} \qquad O(h^2) \qquad (6.7)$$

здесь h – шаг табуляции.

При вычислении производной численными методами возникают ошибки двух видов: ошибки усечения и ошибки округления.

Ошибка усечения пропорциональна величине шага для первой производной и квадрату величины шага для второй производной. С уменьшением шага ошибка усечения уменьшается, но одновременно возрастает ошибка округления. Чтобы избежать получения ошибочных результатов, необходимо контролировать, чтобы разность двух смежных значений функции не была меньше точности вычислений. Например, на компьютере вычисляющем с точностью до 14 знаков, разность не должна быть меньше 10^{-14}.

6.5.2. ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННОГО ИНТЕГРАЛА ЧИСЛЕННЫМИ МЕТОДАМИ

Известно, что определенный интеграл равен площади, ограниченной подынтегральной функцией, осью x-ов и вертикальными прямыми x=a и x=b, где a и b – границы интервала интегрирования функции (см. раздел 3.3.3).

При численном интегрировании интеграл заменяют суммой конечного числа элементарных площадок, вычисленных тем или иным способом:

$$\int_a^b f(x)dx = \sum_{i=1}^{n} h(f(x)), \qquad (6.8)$$

здесь h - шаг табулирования функции.

Поэтому в Excel определенный интеграл легко вычислить протабулировав выражение под знаком суммы (6.8) и подсчитав сумму значений функции на отрезке табулирования. Другой путь – создать функцию пользователя (см. раздел 6.8.2).

6.5.3. ОПРЕДЕЛЕНИЕ КОЭФФИЦИЕНТОВ ЭМПИРИЧЕСКИХ ФОРМУЛ МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ

Нередко при обработке результатов наблюдений встречаются со следующей задачей: в результате проведенных опытов получен ряд значений переменных x и y, однако характер функциональной зависимости между ними остается неизвестным. Требуется по полученным данным найти аналитическое выражение зависимости между ними. Формулы, полученные в результате решения задач подобного рода, называются эмпирическими.

Задача о построении эмпирической формулы состоит в следующем.

Пусть в результате экспериментов получены данные, представленные таблицей:

x_1	x_2	...	x_k	...	x_n
y_1	y_2	...	y_k	...	y_n

где x_i - значения аргументов, изменяющихся с постоянным шагом и расположенных в порядке возрастания их значений, y_i - экспериментальные значения функции, соответствующие данным значениям аргументов.

Требуется найти эмпирическую формулу $y=f(x,a_1,a_2,...a_m)$, где функция f зависит не только от значения аргумента x_i, но и от некоторых ***параметров*** a_j. Разности

$$f(x_i,a_1,a_2,...a_m)-y_i=e_i, \qquad (i=1,2,3,..m) \qquad (6.9)$$

называются отклонениями или погрешностями, где

x_i - числа из первой строки таблицы,

y_i - числа из второй строки данной таблицы,

$f(x_i,a_1,a_2,...a_m)$ - значения функции при соответствующих значениях аргумента x_i.

Параметры a_j эмпирической формулы $y=f(x,a_1,a_2,...a_m)$ необходимо подобрать таким образом, чтобы отклонения e_i оказались наименьшими. Наиболее распространенным критерием является критерий, лежащий в основе ***метода наименьших квадратов***: параметры функции выбирают так, чтобы сумма квадратов отклонений оказалась минимальной:

$$S = \min \sum_{i=0}^{n} e_i^2 = \sum_{i=0}^{n} [f(x_i, a_1, a_2, ..) - y_i]^2 \qquad (6.10)$$

Минимум функции находят, приравнивая нулю частные производные по переменным параметрам a_j:

$$\frac{dS}{da_1} = 0 \ , \ \frac{dS}{da_2} = 0 \ , \ ... \qquad (6.11)$$

Полученные соотношения образуют систему уравнений, для определения коэффициентов a_i, для i=1,2,...,m.

Пусть функция f(x) является многочленом степени m, т.е.

$f(x)=a_0x^m + a_1x^{m-1} + ...+a_ix^{m-i} +...+a_{m-2}x^2\ a_{m-1}x + a_m$, ($a_0$ не равно 0).

Задача ставится следующим образом: подобрать коэффициенты многочлена так, чтобы сумма квадратов отклонения для данного многочлена оказалась минимальной.

В случае m=1 имеем линейное приближение функции по методу наименьших квадратов, в случае m=2 - квадратичное приближение.

В случае линейной зависимости предполагается, что все точки лежат на некоторой прямой

$$y=a^*x+b, \qquad (6.12)$$

где a и b - некоторые постоянные параметры, подлежащие определению. Для их нахождения требуется решить систему уравнений:

$$\begin{cases} a^*\Sigma x_i^2 + b^*\Sigma x^i = \Sigma x^i{}^*y^i \\ a^*\Sigma x_i + b^*n = \Sigma y_i \end{cases} \qquad (6.13)$$

В случае квадратичной зависимости предполагается, что все точки лежат на некоторой параболе. В этом случае естественно предположить, что между ними существует квадратичная зависимость, т.е.

$$y=a^*x^2+b^*x+c, \qquad (6.14)$$

где a, b, c - постоянные параметры, подлежащие определению. Они находятся из системы уравнений:

$$\begin{cases} a^*\Sigma x_i{}^4 + b^*\Sigma x_i^3 + c^*\Sigma x_i^2 = \Sigma\ x_i^2\ {}^*y_i \\ a^*\Sigma x_i^3 + b^*\Sigma x_i^2 + c^*\Sigma x_i = \Sigma x_i{}^*y_i \\ a^*\Sigma x_i^2 + b^*\Sigma x_i + c^*n = \Sigma y_i \end{cases} \qquad (6.15)$$

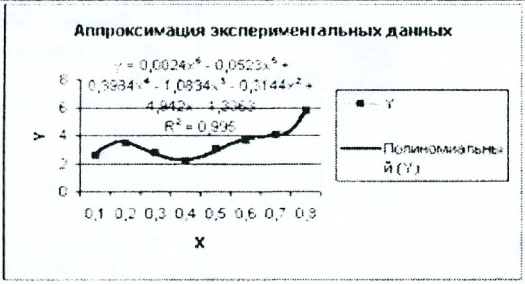
Решение системы осуществляется любым известным методом.

После того как найдены значения коэффициентов систем уравнений (6.13) и (6.15), вычисляют значения выражений (6.12) и (6.14) при заданных значениях аргументов. Каждое из полученных уравнений удовлетворяет условию (6.10).

Для выбора предпочтительной функции из двух также применяют метод наименьших

Листинг 6.11. Определение коэффициентов эмпирических формул с помощью линий тренда

X	Y
0,1	2,56
0,2	3,45
0,3	2,78
0,4	2,12
0,5	2,98
0,6	3,65
0,7	4,01
0,8	5,8



квадратов. Для этой цели находят отклонения по формулам (6.9) и квадраты этих отклонений для каждого метода (6.10). Предпочтительной будет та функция, для которой сумма квадратов отклонений будет наименьшая.

Из приведенного алгоритма видно, что нахождение коэффициентов аппроксимирующей функции достаточно трудоемкая задача. К счастью, электронная таблица Excel имеет ряд технологий для решения подобных задач: использование линий тренда на графиках функций и использование встроенных функций.

Подбор аппроксимирующей функции по графику

Коэффициенты эмпирических формул можно определить подбирая вид аппроксимирующей функции по графику:

- составьте таблицу значений экспериментальных данных X и Y (Листинг 6.11);
- постройте график функции (точечный или график);
- щелкните правой кнопкой мыши по линии графика – открывается контекстное меню;
- выберите в этом меню команду ***Добавить линию тренда***. Открывается окно диалога Линия тренда (рис. 6.14);
- выберите в этом окне подходящую функцию, в данном примере полиномиальную, и выберите степень функции таким образом, чтобы она наиболее точно описывала экспериментальные данные. Достоверность аппроксимации оценивается коэффициентом r^2, где r^2 - коэффициент детерминации является квадратом коэффициента корреляции (r). Он может принимать значения от 0 до 1. Чем больше этот коэффициент, тем ближе располагаются точки линии тренда к экспериментальным точкам на графике. Приближение считается хорошим, если r^2 больше 0,9. Если $R^2=1$, то это означает полное совпадение прогнозируемых и фактических данных;
- откройте закладку ***Параметры*** в окне диалога Линия тренда и установите флажки *Показывать уравнения на диаграмме* и *Поместить на диаграмму величину достоверности аппроксимации* (R^2).

Использование встроенных функций Excel для определения коэффициентов эмпирических формул

Электронная таблица Excel располагает встроенными средствами для определения коэффициентов эмпирических фор-мул – это функции ЛИНЕЙН, ЛГРФПРИБЛ, ТЕНДЕНЦИЯ, РОСТ. Все эти функции возвращают множество точек аппроксимирующей кривой. Функции ЛИНЕЙН и ЛГРФПРИБЛ возвращают, кроме того, и коэффициенты уравнений регрессии.

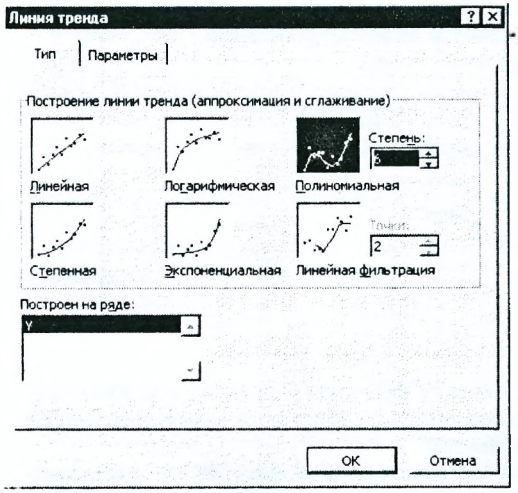


Рис. 6.14. Окно диалога Линии тренда

Для поиска коэффициентов эмпирических формул можно использовать и возможности пакета Анализ.

Функция **ЛИНЕЙН** использует модель многомерной линейной регрессии $y(x_{1,i}, x_{2,i}, \ldots) = A + Bx_{1,i} + C x_{2,i} + \ldots$

Функция имеет следующий синтаксис:

ЛИНЕЙН(Y - массив; X – массив; конст; статистика).

Здесь **Y – массив** – ссылка на массив данных *Y*;

X – массив – ссылка на один или несколько массивов данных *x*;

конст – логическое значение, определяющее константу сдвига. Она может принимать два значения: ИСТИНА (1) и ЛОЖЬ (0). Если *конст* равна 1, то коэффициент A вычисляется обычным образом, иначе коэффициент A будет равен 0 для функции ЛИНЕЙН и 1 для функции ЛГРФПРИБЛ;

статистика – логическое значение, которое указывает, требуется ли вернуть дополнительную статистику регрессии: *стандартная ошибка коэффициентов, стандартная ошибка оценки y, число степеней свободы, F-статистика, Регрессионная сумма квадратов и Остаточная сумма квадратов.*

Стандартная ошибка для оценки *y* вычисляется по формуле

$$S_{y-x} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - y(x_i))^2}{p}},$$

где *p* – число степеней свободы, y_i - заданное значение функции, $y(x_i)$ – вычисленное значение функции.

Стандартные ошибки коэффициентов S_A и S_B вычисляются по формулам

$$S_A = \sqrt{\frac{1}{n} + \frac{\langle x \rangle^2}{\sum_{i=1}^{n} (x_i - \langle x \rangle)^2}} S_{y-x}, \text{ где } \langle x \rangle = \frac{\sum_{i=1}^{n} x_i}{n}.$$

Число степеней свободы p равняется разности числа точек и числа коэффициентов в уравнении регрессии. Например, уравнение прямой линии имеет два коэффициента, соответствующие углу наклона прямой и сдвигу по оси *y*. Если число заданных точек равно 10, то число степеней свободы будет равно 8.

F-статистика – используется совместно с F-значениями для оценки вероятности того, что данные действительно описываются данными функциями, а не вызваны случайными флуктуациями. F-значения вычисляются с помощью функции FРАСПОБР(). Если

значения F-статистики больше соответствующего значения F, то это означает, что совпадение обусловлено реальной корреляцией, а не случайными флуктуациями.

Коэффициент детерминированности вычисляется по формуле:

$$r^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - y(x_i))^2}{\sum_{i=1}^{n} (y_i - \langle y_i \rangle)^2}$$

Регрессионная сумма квадратов вычисляется по формуле:

$$\sum_{i=1}^{n} (y_i - \langle y_i \rangle)^2, \text{ где } \langle y_i \rangle = \left(\sum_{i=1}^{n} y_i \right) / n$$

Остаточная сумма квадратов (сумма наименьших квадратов) вычисляется по формуле:

$$\sum_{i=1}^{n} (y_i - y(x_i))^2$$

Функция **ЛГРФПРИБЛ** реализует следующую модель:

$y(x_1,x_2,...)= A(B^{x_1})(C^{x_2}) ...$

Синтаксис функции:

ЛГРФПРИБЛ(Y - массив; X – массив; конст; статистика)

Рассмотрим использование этих возможностей на примерах.

Использование функции ЛИНЕЙН

Оформите таблицу регрессии в соответствии с Листингом 6.12.

Введите в столбцы A и B экспериментальные значения X и Y.

Листинг 6.12. Линейная аппроксимация

	A	B	C	D	E	F	G	H
	Линейная аппроксимация							
1	***X***	***Y***	***Ожидаемое***					
2	250	0,445	0,36156044		Таблица регрессии			
3	300	0,362	0,336313187			B	A	Комментарий
4	350	0,302	0,311065934			-0,00050	0,487797	Коэффициенты
5	400	0,256	0,285818681		S_A, S_B	0,00006	0,033366	Стд. Ошибка коэфф.
6	450	0,223	0,260571429		r^2	0,87542	0,038741	Коэфф. детерминации
7	500	0,197	0,235324176		F	77,29486	11	Степени свободы
8	550	0,176	0,210076923		Сумма кв.	0,11601	0,01651	
9	600	0,158	0,18482967			регресси-онная	остаточная	

Введите в ячейки F3 и G3 обозначение коэффициентов регрессии - символы "B" и "A". Присвойте ячейкам F4 и G4 имена B и A, соответственно (чтобы не использовать абсолютный адрес при копировании формул).

В ячейку C2 введите формулу B*A2 + A (то есть используется простейшая формула линейной аппроксимации y = Bx + A) и скопируйте эту формулу в соответствующие ячейки колонки C.

Листинг 6.13. Логарифмическое приближение

	A	B	C	D	E	F	G	H	I
1		**Логарифмическое приближение**							
2	***X1***	***X2***	***Y***	***Ожидаемое***					
3	250	1	0,445	,3833		**Таблица регрессии**			
4	300	1,5	0,362	,3408		C	B	A	
5	350	2	0,302	,3031		0,338465	1,0085	0,135841	Коэффициенты
6	400	2,5	0,256	,2695		0	0	0	Станд. Ошибка коэфф.
7	450	3	0,223	,2396	r^2	0,975031	0,08022	#Н/Д	Стд. ошибка оценки Y.
8	500	3,5	0,197	2130	F	195,2452	10	#Н/Д	Степени свободы
9	550	4	0,176	,1894	Суммы кв.	2,51316	0,06435	#Н/Д	
10	600	4,5	0,158	,1684		↑	↑		
11	650	5	0,144	,1497		регрессионная	остаточная		
12	700	5,5	0,132	,1331					

Выделите блок F4:G8. Введите в первую ячейку выделенного блока функцию ЛИНЕЙН(B2:B9;A2:A9,1,1) и вставьте ее во весь блок комбинацией клавиш Ctrl+Shift+Enter.

Результат: y=-0,00050*x+0,487797

Имеет место достаточное высокое совпадение результатов регрессионного анализа с исходными данными, так как R^2=0,87542.

Использование функции ЛГРФПРИБЛ

Оформите таблицу регрессии в соответствии с листингом 6.13.

Функция логарифмического приближения применяется аналогично функции линейного приближения. Если имеется два вектора ***X,*** то в качестве блока аргумента *x* указывать область A2:B12. Векторы X1 и X2 не должны совпадать.

Введите в столбцы A, B и C заданные значения X1 и X2 и Y.

Введите в ячейки F3 и G3 обозначение коэффициентов регрессии - символы "C", "B" и "A". Присвоить ячейкам F5, G5 и H5 имена C, B и A, соответственно (чтобы не использовать абсолютный адрес при копировании формул).

В ячейку C2 введите формулу $A(B^{x_1})(C^{x_2})$, то есть ограничимся двумя векторами **X,** и скопируем эту формулу в соответствующие ячейки колонки D.

Выделите блок F5:H9. Введите в первую ячейку выделенного блока функцию ЛГРФПРИБЛ(C3:C12;A3:B12,1,1) и вставьте ее во весь блок.

Результат: $y=0{,}135841*(1{,}0085^{x1})*(0{,}338465^{x2})$

Степенная регрессия

Функции Excel не рассчитаны на выполнение степенной регрессии, но функцию ЛИНЕЙН можно легко приспособить для вычисление коэффициентов эмпирических формул с использованием степенной регрессии $y=A + Bx +Cx^2+$ ... Для этого в выражении множественной регрессии вводят следующие замены: $x_{1,i}=x_i$; $x_{2,i}=x_i^2$; $x_{3,i}=x_i^3$…

Пример использования степенной регрессии приведен на Листинге 6.14.

Листинг 6.14. Степенная регрессия

	A	B	C	D	E	F	G	H
1		**Степенная регрессия**						
2	X	x^2	X^3	Y	Ожидаемое			
3	250	62500	15625000	0,445	0,4399478			
4	300	90000	27000000	0,362	0,3661319			
5	350	122500	42875000	0,302	0,3062602			
6	400	160000	64000000	0,256	0,2586284			
7	450	202500	91125000	0,223	0,2215317			
8	500	250000	125000000	0,197	0,1932657			
9	550	302500	166375000	0,176	0,1721259			
10	600	360000	216000000	0,158	0,1564076			
11	650	422500	274625000	0,144	0,1444063			
12	700	490000	343000000	0,132	0,1344176			
13	750	562500	421875000	0,121	0,1247368			
14	800	640000	512000000	0,112	0,1136593			
15	850	722500	614125000	0,103	0,0994808			
16								
17		**Таблица регрессии**						
18		D	C	B	A	Комментарий		
19		-2E-09	4,83432E-06	-0,0036	1,0778511	Коэффициенты		
20		2E-10	3,59808E-07	0,00019	0,0303105	Стд. Ошибка коэфф.		
21	r^2	0,999	0,003893183	#Н/Д	#Н/Д	Стд. ошибка оценки Y.		
22	F	2911,4	9	#Н/Д	#Н/Д	Степени свободы		
23	Сумма кв.	0,1324	0,000136412	#Н/Д	#Н/Д			

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте постановку задачи для определения коэффициентов эмпирических формул методом наименьших квадратов.
2. В чем заключается суть метода наименьших квадратов?
3. Запишите уравнения линейной и квадратичной зависимостей.
4. Как построить графики функций линейной и квадратичной зависимостей?
5. Как построить линию тренда и вывести на график уравнение регрессии и значение коэффициента детерминированности?
6. Какие функции Excel могут использоваться для определения коэффициентов эмпирических формул.
7. Запишите уравнения регрессии для линейной и логарифмической функций.
8. Запишите уравнение квадратичной (степенной) регрессии и поясните, как ее можно реализовать с помощью функции ЛИНЕЙН.

ЗАКЛЮЧЕНИЕ

Программа Excel имеет большое число встроенных функций, а также специальный Пакет анализа для обработки данных. В настоящем разделе мы познакомились с некоторыми возможностями программы. Для определения коэффициентов эмпирических формул по данным, представленным в виде таблицы зависимости функции от аргумента, могут использоваться функции ЛИНЕЙН, ЛГРФПРИБЛ, а также графические средства. В частности линии тренда.

6.6. РЕШЕНИЕ АЛГЕБРАИЧЕСКИХ И ТРАНСЦЕНДЕНТНЫХ УРАВНЕНИЙ

6.6.1. МЕТОДЫ, ОСНОВАННЫЕ НА ТАБУЛИРОВАНИИ ФУНКЦИЙ

Процесс нахождения корней состоит, как известно, из двух этапов: *отделения корней* и *уточнения значения корней* на отрезках отделения с заданной точностью (см. раздел. 3.3.3).

Отделением корней называется процесс выделения из области определения функции f отрезков [a,b], в каждом из которых содержится один и только один корень уравнения f(x)=0.

Корни уравнения могут находиться на интервалах, определяемых переменой знака функции, между критическими точками. К критическим относятся точки, в которых производная от функции f(x) обращается в нуль, а также граничные точки.

В электронной таблице отделение корней можно выполнить путем табулирования функции с некоторым, достаточно малым, шагом. Областями отделения корней будут значения аргументов, между которыми происходит смена знака функции, а также графическим методом: - построить график функции на заданной области определения функции и выделить отрезки, на которых функция меняет знак.

Под **уточнением значения** корня с заданной точностью h понимают сужение границ отрезка [a,b] до длины, не превосходящей h. Уточнение значения корня на отрезке отделения осуществляется различными методами одномерной поисковой оптимизации: простых итераций, деления отрезка пополам, касательных, хорд, наискорейшего спуска и др.

Для уточнения значения корня в Excel могут применяться все названные методы. В настоящем пособии рассмотрены три метода:

- простое табулирование;
- метод простых итераций;
- метод Ньютона (метод касательных).

Метод простого табулирования не имеет ограничений, но малоэффективен, требует большого числа ручных операций. Условием окончания процедуры поиска является достижение функцией f(x) заданного значения: $f(x_i)<=e$, - где e - заданные требования к точности поиска корня.

Метод простых итераций более эффективен. Он сходится, если $f'(x)<0$. Для его реализации необходимо функцию f(x)=0 преобразовать к рекуррентному виду

$$x_{i+1}=\varphi(x_i) \qquad (6.16)$$

Табулировать необходимо правую часть выражения (6.16). Для первой формулы в качестве аргумента используется начальное приближение X_0 (как правило, одна из границ отрезка отделения), для последующих формул - значение корня на предыдущем шаге, т. е. $f(x_i)$. Начальное приближение выбирается произвольно. Условие окончания процедуры вычисления $\varphi(x_{i+1})- \varphi(x_i)<=e$.

Метод Ньютона самый эффективный метод. Он обеспечивает сходимость за минимальное число шагов. Однако этот метод накладывает серьезные ограничения на вид функции. Функция должна быть дважды дифференцируема. Для поиска корня в этом методе, так же как и в методе простых итераций, составляется рекуррентная формула:

$$x_{i+1}=x_i - f(x_i)/f'(x_i). \qquad (6.17)$$

Табулировать необходимо правую часть выражения. Начальное приближение выбирается на одной из границ отрезка отделения корня. В качестве начального приближения x_o выбирается граница ***b***, если $f'(x)*f''(x)>0$, и граница ***a***, если $f'(x)*f''(x)<0$. Для первой формулы в качестве аргумента используется начальное приближение x_o, для последующих формул - значение корня на предыдущем шаге, т. е. x_i. Условием окончания процедуры уточнения корня является достижение функцией значения меньше заданного - $f(x_{i+1})<=e$.

Приближенное значение производных можно вычислить численно с помощью формул (6.6), (6.7):

Значение Δx можно принять в интервале от 0.0001 до 0.00001

Все указанные методы могут быть реализованы с помощью функций пользователя, разработанных с помощью встроенного языка программирования Visual Basic for Application (VBA).

Пример 6.6. Пример отделения и уточнения корня

Пусть требуется найти корни уравнения $y=2^x+2x-5$ с точностью 0,001.

Загрузите электронную таблицу.

1. Отделите корни уравнения. Для этой цели протабулируйте функцию на значительном отрезке с большим шагом и зафиксируйте границы смены знака функции. После нескольких шагов получим отрезок отделения [1.28 – 1.285]. Выберите шаг 0,0005 и протабулируейте функцию вновь. Получен новый интервал, где функция меняет свой знак [1,283; 1,2835] (Листинг 6.15). На границе отрезка при x=1.283 значение функции равно 0,000555. То есть функция меньше заданной точности. Эту точку и примем в качестве значения корня заданной функции. ***Результат***: x=1.283, f(x)=0,000555.

2. Уточните значение корня на отрезке отделения [1.283; 1.2835] методом итераций:

- напишите рекуррентную формулу. В заданном выражении это обеспечивается просто путем переноса второго слагаемого в левую часть уравнения и делением на коэффициент при неизвестной:

$$x=(5-2^x)/2 \qquad (6.18)$$

- запишите в ячейку C11 формулу (6.18). В качестве x примите значение функции на левой границе отрезка отделения, то есть 1.28 (ячейка D4);

Листинг 6.15. Решение алгебраических и нелинейных уравнений

	A	B	C	D	E	F
1	***Решение нелинейных уравнений y=2^x+2x-5***					
2						
3	1. Простое табулирование		2. Метод итераций		3. Метод Ньютона	
4	Начальное значение	0	Нач. знач.	1,28	Нач. знач.	1,28
5	Шаг табуляции	0,0005	Шаг таб.	нет	Шаг таб.	нет
6	Точность	0,001	Точность	0,001	Точность	0,001
7	Формула y=2^x+2x-5		y=2^x+2x-5		y=2^x+2x-5	
9			Рекуррентная формула x=(5-2^x)/2		x=x-(2^x+2x-5)/ (2^x* LN(2)+2)	
10	X	Функция f(x)	Функция $\varphi(x_i)$	Разность ABS(X_{i+1} - x_i)	Функция x_i- $f(x_i)/f'(x_i)$	Функция f(x)
11	1.28	-.011610	1,285805		1,80719	2,11396
12	1.2805	-.009768	1,28091	0,0048955	1,32953	0,17228
13	1.281	-.007926	1,285039	0,0041297	1,28349	0,00127
14	1.2815	-.006084	1,281557	0,0034828	1,28315	0,00000
15	1.282	-.004241	1,284494	0,0029379		
16	1.2825	-.002399	1,282017	0,0024778		
17	1.283	-.000555	1,284107	0,0020901		
18	1.2835	.0012882	1,282344	0,0017628		
19	1.284	.0031320	1,283831	0,0014869		
20	1.2845	.0049761	1,282577	0,0012541		
21	1.285	.0068205	1,283635	0,0010578		
22			1,282742	0,0008922		

- запишите в ячейку C12 формулу (6.18), но в качестве x примите значение ячейки C11, то есть значение корня на предыдущем шаге;

- запишите в ячейку D12 формулу ABS(x_i-x_{i+1});

- скопируйте формулы из ячеек C12, D12 в нижележащие ячейки, пока в колонке D не будет выполнено требование к разности двух смежных значений аргументов.

Результат: x=1.282742, f(x)= 0,00089.

3. Уточните значение корня на отрезке отделения [1.283; 1.2835] методом Ньютона:

- найдите первую производную для заданного выражения

$$f'(x)=(2^* 2^x * LN(2)+2) \quad (6.19)$$

$$x_{i+1}=x_i-(2^x x_i+2x_i-5)/ (2^* 2^x x_i^* LN(2)+2) \quad (6.20)$$

- запишите в ячейку E11 формулу (6.20). В качестве начального значения x примите значение функции на левой границе отрезка отделения, то есть 1,28 (ячейка F4);

- запишите в ячейку E12 формулу (6.20), но в качестве x примите значение ячейки E11, то есть значение корня на предыдущем шаге;

- запишите в ячейку F11 формулу ($f(x_i)$, со ссылкой на ячейку E11;

- скопируйте формулы из ячеек E12 и F11 в нижележащие ячейки, пока в колонке F не будет выполнено требование к точности значения функции.

Результат: x=1,283141; f(x)= -3,5E-05

Из приведенных примеров наглядно видно, что метод Ньютона (метод касательных) обеспечивает гораздо более быструю сходимость, чем метод простых итераций.

6.6.2. ИСПОЛЬЗОВАНИЕ ВСТРОЕННЫХ ПРОЦЕДУР

Другим способом решения линейных и нелинейных уравнений является использование возможностей программы Excel по оптимизации решений. Для этой цели служат команды ***Подбор параметра*** и ***Поиск решения*** меню ***Сервис***. Если команды Поиск решения нет в меню Сервис, то ее необходимо загрузить командой ***Сервис***, ***Надстройки***. Функция Поиск решения может использоваться для решения различных задач оптимизации в том числе и задач линейного программирования.

Использование возможностей Excel по оптимизации рассмотрим на примерах.

Пример 6.7. Решим квадратное уравнение $3^*x^2+2x-15$ с точностью 0,0001.

Результаты решения обоими методами приведены на Листинге 6.16.

Листинг 6.16. Решение уравнений

	A	B	C
8		**Исходное состояние**	
9		**Подбор параметра**	
10		Функция	Корень
11	1	-4,0000E+00	-1,0000
12	2	6,0000E+01	-5,0000
13	3	8,0000E+01	5,0000
14			
15		**Поиск решения**	
16		Функция	Корень
17	1	-4,0000E+00	-1,0000
18	2	6,0000E+01	-5,0000
19	3	8,0000E+01	5,0000
20	4	6,0000E+01	-5,0000

	A	B	C
		Результат решения	
9		**Подбор параметра**	
10		Функция	Корень
11	1	-8,4213E-04	-1,6666
12	2	4,9306E-04	-1,6667
13	3	1,5199E-05	1,0000
14			
15		**Поиск решения**	
16		Функция	Корень
17	1	3,2259E-07	-1,6667
18	2	-4,1624E-07	-1,6667
19	3	0,0000E+00	1,0000
20	4	0,0000E+00	1,0000

Порядок решения уравнения с использованием функции Подбор параметра

Введите в ячейку B11 формулу 3*C11^2+2*C11-15. В ячейку C11 введите начальное значение x=-1. Результат приведен на листинге в столбцах B и C.

Введите команду ***Сервис***, ***Подбор параметра*** – открывается одноименное окно диалога (рис. 6.15).

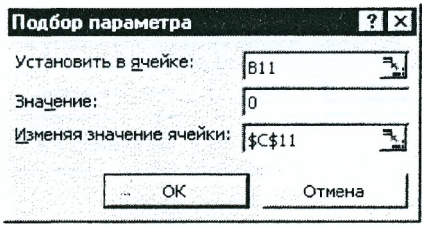


Рис.6.15. Окно диалога функции Подбор параметра

Введите в строку "Установить в ячейке" адрес ячейки, содержащей формулу - B11, в строку "Значение" введите значение "0", а в строку "Изменяя значение ячейки" - адрес ячейки, содержащий значение X - C11 с абсолютным адресом. Щелкните по кнопке OK. В ячейке B11 отобразится значение точности поиска решения, а в ячейке C11- значение корня. Для сравнимости результатов на Листинге 6.16 в левой таблице показано исходное состояние, а в правой - результаты. Результат решения зависит от начального значения корня. Сравните результаты в строках

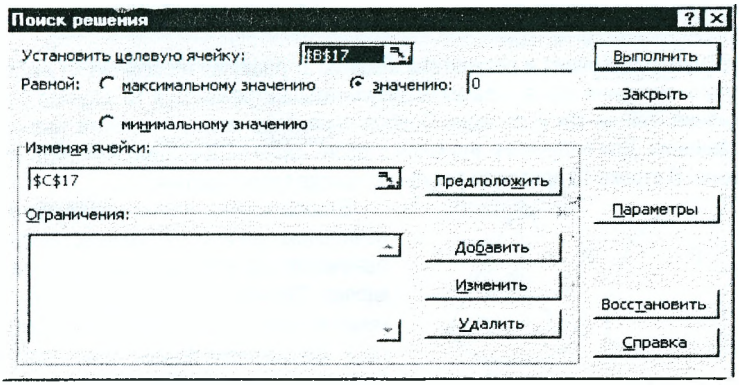


Рис.6.16. Окно диалога функции Поиск решения

11 и 12. Функция Подбор параметра позволяет найти только один корень уравнения. Для получения значения второго корня необходимо изменить начальное значение. То есть, предварительно надо исследовать функцию, например, графическим методом и отделить корни уравнения. В строке 13 приведен результат поиска второго корня.

Порядок решения уравнения с использованием функции Поиск решения

Введите в ячейку B17 формулу 3*C11^2+2*C11-15. В ячейку C17 введите начальное значение x=-1. Результат приведен на листинге 6.16 в столбцах B и C.

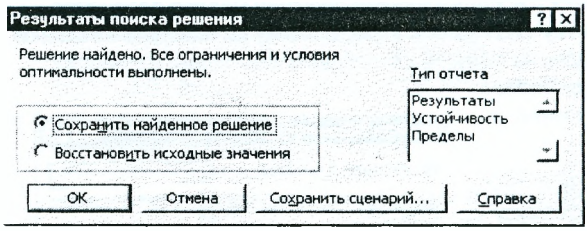


Рис. 6.17. Окно диалога "Результаты поиска решения"

Введите команду ***Сервис***, ***Поиск решения*** – открывается одноименное окно диалога (рис. 6.16).

Введите в строку "Установить целевую ячейку" адрес ячейки, содержащей формулу - B17, установите переключатель "Значение" и в строку вода "значению" введите значение "0", а в строку "Изменяя ячейки" введите адрес ячейки содержащий значение X - C17 с абсолютным адресом. Щелкните по кнопке Выполнить. Открывается окно "Результаты поиска решения" (рис. 6.17). Для получения результата щелкните по кнопке OK. В ячейке B17 отобразится значение точности поиска решения, а в ячейке C17- значение корня. Для сравнимости результатов на Листинге 6.16 приведены исходные состояния и результаты. Результат решения не зависит от выбранного начального значения корня. Сравните результаты в строках 17 и 18. Однако Функция Поиск решения, так же как и функция Подбор параметра позволяет найти только один корень уравнения. Для получения значения второго корня необходимо изменить начальное значение. В строке 19 приведен результат поиска второго корня.

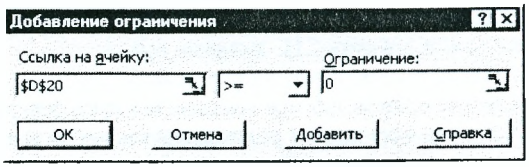


Рис. 6.18. Окно диалога "Добавление ограничения"

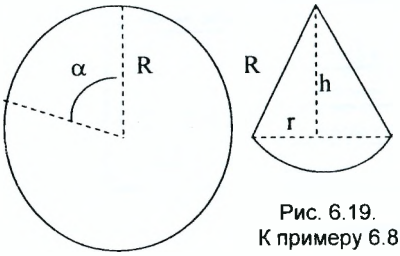


Рис. 6.19. К примеру 6.8

Второй корень можно найти при любом начальном значении X, если установить ограничения на его значение. Щелкните по кнопке ***Добавить*** в окне диалога Поиск решения (рис. 6.16). Открывается окно диалога "Добавление ограничения" (рис. 6.18). Выберите в списке "Ссылка на ячейку" адрес ячейки, содержащей значение x, например, D20, в среднем списке - знак отношения, а в списке "Ограничения" введите значение ограничения, например, "0" и щелкните по кнопке OK. Результат будет записан в окне "Ограничения" окна диалога Поиск решения (рис.6.16). Пример решения приведен в строке 20 Листинга 6.16.

Из анализа результатов на Листинге 6.16. можно сделать следующие выводы: Функция Поиск решения дает более точные результаты по сравнению с функцией Подбор параметра, при этом результат не зависит от начального приближения. Функция может найти только одно решение, поэтому поиск значения корня необходимо вести на отрезке отделения.

Пример 6.8. Задача о "пожарном ведре". Дана заготовка из жести в виде круга диаметром R=0,75 м. Требуется выкроить из него конусообразное ведро таким образом, чтобы объем ведра был наибольший.

Разработаем математическую модель:

Объем пожарного ведра Q=1/3hSосн, Sосн= πr^2,

где r – радиус основания конуса, h – высота ведра, h=корень(R^2-r^2)

Радиус основания зависит от угла вырезки α. Длина окружности основания ведра $L=(2\pi-\alpha)R$ или $L=2\pi r$, отсюда

$$r=(2\pi-\alpha)R/(2\pi),$$

$$Q=1/3*Корень(R^2-((2\pi-\alpha)R/(2\pi))^2)*\pi*(2\pi-\alpha)R/(2\pi))^2 \qquad (6.21)$$

Объем ведра Q должен максимизироваться, поэтому выражение (6.21) называют целевой функцией. Оптимизируемым параметром является угол α. Ограничение в данном выражении одно: угол α должен быть больше 0, но меньше 2π. Решение приведено на Листинге 6.17. Окно диалога приведено на рис. 6.20.

Листинг 6.17. Оптимизация

Задача о пожарном ведре	
Q=1/3*Корень(R^2-(2ПИ)R/ (2П))*П*(2ПИ)R/ (2П))^2	
Исходные данные:	
R=	0,75
Угол альфа	Целевая функция
1,84	0,046875

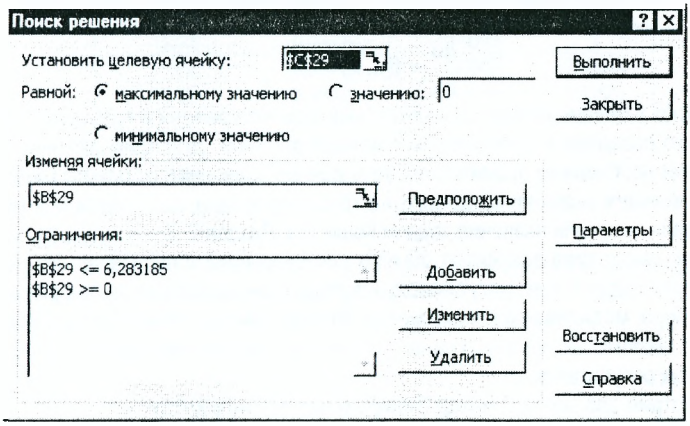


Рис.6.20. Окно диалога Поиск решения при решении задачи о "Пожарном ведре"

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем отличие алгебраического уравнения от трансцендентного?
2. Что называется корнем уравнения?
3. Что такое отделение корня? Для какой цели оно производится?
4. Что такое уточнение значения корня, какими методами оно осуществляется?
5. Поясните, в чем заключается метод простых итераций?

6. Поясните принцип использования метода касательных для уточнения значения корня.

7. Что такое рекуррентная формула, как она получается?

8. Какие команды Excel могут использоваться для решения линейных и нелинейных уравнений?

9. Поясните порядок решения уравнения с помощью модуля Подбор параметра?

10. Поясните порядок решения уравнений с помощью модуля Поиск решения?

ЗАКЛЮЧЕНИЕ

Электронная таблица имеет встроенные процедуры для решения нелинейных и линейных уравнений: Подбор параметра и Поиск решения. Однако эти процедуры находят только одно решение, ближайшее к заданному начальному приближению. Поэтому функцию необходимо предварительно исследовать и отделить корни уравнения, например, путем табулирования или построения графика функции.

Процедуры Подбор параметра и Поиск решения позволяют также решать задачи одномерной поисковой оптимизации.

6.7. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА С НАЧАЛЬНЫМИ УСЛОВИЯМИ

6.7.1. ПОСТАНОВКА ЗАДАЧИ

Пусть некоторый процесс описывается дифференциальным уравнением первой степени.

$$\frac{du}{dx} = f(u, x) \qquad (6.22)$$

Известно значение процесса в некоторый момент времени и требуется оценить значение этого процесса в произвольный момент времени. Для приближенного решения этой задачи необходимо проинтегрировать данное уравнение от одного конца с известными граничными значениями до другого конца интервала, на котором они неизвестны. Такие задачи получили название задачи Коши (см. Раздел 2.2).

Задачи такого типа возникают обычно для уравнений с производными по времени. Для решения задач Коши могут использоваться разные методы: метод рядов Тейлора, метод Эйлера, модифицированный метод Эйлера и метод Рунге-Кутта. Эти методы отличаются сложностью используемых выражений, скоростью сходимости и точностью получаемых результатов.

Рассмотрим два метода: метод Эйлера и метод Рунге-Кутта.

6.7.2. МЕТОД ЭЙЛЕРА

В методе Эйлера значение функции на следующем шаге вычисляется в соответствии с выражением:

$$u(x+h)=u(x)+h*u'(x) \qquad (6.23)$$

Пример 6.6. Найти решение дифференциального уравнения:

$$(1+x^2)^{1/2}\frac{du(x)}{dx} + u(x) = x \text{ при } x>0 \qquad (6.24)$$

при x=0,2 с начальным условием u(0)=0.

Решение:

Приведем выражение (6.24) к виду выражения (6.22):

$$\frac{du(x)}{dx} = (x - u(x)) / \sqrt{(1 + x^2} \qquad (6.25)$$

тогда согласно 6.23

$$u(x + h) = u(x) + h * (x - u(x)) / \sqrt{(1 + x^2)} \qquad (6.26)$$

Теперь для получения результата достаточно сгенерировать ряд значений аргумента x с некоторым шагом и протабулировать правую часть выражения (6.25) на отрезке от 0 до 0,2.

Для примера (6.24) известно аналитическое выражение для вычисления значения функции в произвольной точке:

$$u(x) = 0{,}5(x - \ln(x + \sqrt{1 + x^2}) / (x + \sqrt{1 + x^2})), \qquad (6.27)$$

поэтому можно вычислить относительную погрешность вычисления по формуле

$$\Delta = (u(x)_a - u(x)) / u(x)_a , \qquad (6.28)$$

где $u(x)_a$ – значение функции, вычисленное по аналитической формуле.

Листинг 6.18. Решение дифференциального уравнения методом Эйлера

	A	B	C	D
5	x	y	Аналитика	Погрешность
6	0	0,0000	0,0000	#ДЕЛ/0!
7	0,01	0,0000	0,0000	1
8	0,02	0,0001	0,0002	0,49664
9	0,03	0,0003	0,0004	0,32885
...	...			
24	0,18	0,0144	0,0152	0,04918
25	0,19	0,0160	0,0168	0,04624
26	0,2	0,0177	0,0186	0,04359

Порядок работы:

- внесите в ячейки A6 и B6 (Листинг 6.18) начальные значения аргумента и функции;
- сгенерируйте в столбце A ряд значений аргумента от 0 до 0,2 с шагом 0,01;
- запишите в ячейку B7 правую часть формулы (6.26) со ссылками на адреса ячеек: B6+(A7-A6)*(A6-B6)/КОРЕНЬ(1+A6^2);
- запишите в ячейку C6 правую часть формулы (6.27) со ссылками на адреса ячеек: 0,5*(A6-LN(A6+КОРЕНЬ(1+A6^2))/(A6+КОРЕНЬ(1+A6^2)));
- запишите в ячейку D6 правую часть формулы (6.28) со ссылками на адреса ячеек: (C6-B6)/C6;
- скопируйте формулу из ячеек B7, C6, D6 в нижележащие ячейки.

Результаты:

Методом Эйлера : u(0,2)= 0,0177

По аналитической зависимости: u(0,2)= 0,0186

Погрешность около 4 процентов Δ =0,04359

6.7.3. МЕТОД РУНГЕ-КУТТЫ

Одним из наиболее распространенный методов решения дифференциального уравнения вида y'=f(x,y) на заданном отрезке [Хнач,Хкон] с начальными условиями являет-

ся метод Рунге-Кутты (см. раздел 2.7.2). При решении данного уравнения точное значение y заменяют его приближенным значением:

$$Y_{i+1} = Y_i + (K_1 + 2K_2 + 2K_3 + K_4)/6, \quad (6.29)$$

где значения коэффициентов на i-том шаге вычисляются по формулам:

$$K_1 = h*f(x_i, y_i)$$
$$K_2 = h*f(x_i + 0.5h, y_i + 0.5K_1)$$
$$K_3 = h*f(x_i + 0.5h, y_i + 0.5K_2) \quad (6.30)$$
$$K_4 = h*f(x_i + h, y_i + K_3)$$

Здесь $x_{i+1} = x_i + h$.

Значение шага при переходе к следующей точке можно изменять. Правильность выбора шага проверяется по формуле

$$T = |(K_2 - K_3)/(K_1 - K_2)|, \quad (6.31)$$

величина T не должна превышать нескольких сотых.

Для проверки точности необходимо сделать второй проход с шагом h/2. Если разница между Y_{i+1} и Y_i на k+1 и k проходах будет меньше требуемой точности, то процесс вычисления прекращается. Значение точности выбирают в интервале от 0.001 до 0.00001.

Грубую оценку погрешности метода проводят по формуле

$$Y_k - Y(x_i) = |Y_{k+1} - Y_k|/15, \quad (6.32)$$

где $Y(x_i)$ - значение точного решения уравнения.

Для решения задачи необходимо вычислить значения коэффициентов $K_1 - K_4$ при начальных условиях, выбрать начальный шаг и вычислить значения X_i и Y_i. Затем увеличить значение x и повторить процедуру вычисления.

Пример 6.9. Решить дифференциальное уравнение первого порядка $dY/dx = 2(x^2+Y)$ методом Рунге-Кутты при $0 <= x <= 1$, $Y(0)=1$, шаг $h=0.1$.

Для этой задачи известно аналитическое выражение

$$Y = 1.5e^{2x} - x^2 - x - 0.5. \quad (6.33)$$

Порядок работы

– оформите таблицу согласно Листингу 6.19;

– запишите исходные данные и шапку таблицы в строки 2 и 3;

– сгенерируйте в ячейки A4:A14 значения аргумента;

– запишите в ячейки B4 и G4 начальное значение Y(0);

– запишите в ячейки расчетные формулы со ссылками на ячейки таблицы: коэффициент K1, ячейка C5 - $B$2*2*(A4^2+G4);

коэффициент K2, ячейка D5 - $B$2*2*((A4+$B$2/2)^2+(G4+C5/2));

коэффициент K3, ячейка E5 - $B$2*2*((A4+$B$2/2)^2+(G4+D5/2));

коэффициент K4, ячейка F5 - $B$2*2*((A4+$B$2)^2+(G4+E5));

Y, ячейка G5 - G4+(C5+2*D5+2*E5+F5)/6;

– в ячейку B5 запишите формулу: = G5;

в ячейку I5 запишите формулу (6.31) для контроля выбора шага (D5-E5)/(C5-D5);

– скопируйте формулы из ячеек B5:I5 в нижележащие ячейки.

Листинг 6.19. Решение дифференциального уравнения методом Рунге-Кутты

	A	B	C	D	E	F	G	I
1	Исходные данные							
2	h=	0,1	Xn=	0,00000	Y(0)=	1,00000		Контроль выбора шага
3	Xn=	Yn=	K1=	K2=	K3=	K4=	Yn+1=	
4	0,00	1,0000					1,0000	
5	0,10	1,2221	0,2	0,2205	0,2225	0,2465	1,2221	0,10
6	0,20	1,4977	0,2464	0,2735	0,2762	0,3076	1,4977	0,10
7	0,30	1,8431	0,3075	0,3428	0,3463	0,3868	1,8431	0,10
8	0,40	2,27829	0,3866	0,4318	0,4363	0,4879	2,2783	0,10
9	0,50	2,8274	0,4877	0,5449	0,5507	0,6158	2,8274	0,10
10	0,60	3,5201	0,6155	0,6875	0,6947	0,7764	3,5201	0,10
11	0,70	4,3927	0,7760	0,8661	0,8751	0,9771	4,3927	0,10
12	0,80	5,4894	0,9765	1,0887	1,0999	1,2265	5,4894	0,10
13	0,90	6,8643	1,2259	1,3650	1,3789	1,5357	6,8643	0,10
14	1,00	8,5834	1,5348	1,7069	1,7245	1,9177	8,5834	0,10

Сравнительные результаты вычисления дифференциального уравнения (6.24) методом Рунге-Кутты, методом Эйлера и по аналитической формуле приведены на листинге 6.20. Из таблицы видно, что метод Рунге-Кутты позволяет получить результаты с высокой точностью во всем диапазоне изменения значения аргумента, чего нельзя сказать о методе Эйлера.

Листинг 6.20. Сравнение методов решения дифференциальных уравнений

x	Метод Эйлера	Метод Рунге-Кутты	Точное решение
0,0000	1,0000	1,0000	1,0000
0,1000	1,2000	1,2221	1,2221
0,2000	1,4420	1,4977	1,4977
0,3000	1,7384	1,8432	1,8432
0,4000	2,1041	2,2783	2,2783
0,5000	2,5569	2,8274	2,8274
0,6000	3,1183	3,5201	3,5202
0,7000	3,8139	4,3927	4,3928
0,8000	4,6747	5,4894	5,4895
0,9000	5,7377	6,8643	6,8645
1,0000	7,0472	8,5834	8,5836

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Сформулируйте постановку задачи Коши для решения дифференциального уравнения первого порядка.
2. Поясните порядок решения дифференциального уравнения методом Эйлера.
3. Поясните порядок решения дифференциальных уравнений методом Рунге-Кутты.

ЗАКЛЮЧЕНИЕ

Электронная таблица Excel не имеет встроенных функций для решения дифференциальных уравнений, однако позволяет решать их с использованием классических методов, например, метода Эйлера или Рунге-Кутты. Метод Рунге Кутты, несмотря на его громоздкость, дает более точные результаты. Для получения результатов с заданной точностью необходимо использовать метод двойного прохода.

6.8 АВТОМАТИЗАЦИЯ ВЫЧИСЛЕНИЙ В EXCEL

6.8.1. СОЗДАНИЕ МАКРОСОВ

Для автоматизации выполнения часто повторяющихся операций, например, форматирования выделенных ячеек, оформления шапок таблиц и т. п. в Excel можно создавать макрокоманды – макросы (см. также раздел 5).

Удобнее всего макросы создавать путем записи. Правда, при этом в макрос записываются все действия пользователя, в том числе и ошибочные. Однако, если алгоритм работы продуман, то проблем не возникает.

Порядок создания макроса

Введите команду ***Сервис, Макрос, Начать Запись***. Открывается окно диалога (рис. 6.21), в котором необходимо указать имя макроса, назначить, при необходимости, сочетание клавиш для запуска макроса, указать место хранения макросов и щелкнуть по кн. Ок. После этого необходимо выполнить нужные действия. Во время записи макроса на экране присутствует панель инструментов управления записью макроса с одной кнопкой – Остановка. Для окончания записи макроса щелкните мышью по этой кнопке или введите команду ***Сервис, Макросы, Остановить запись***.

При использовании макросов в Excel необходимо сразу позаботиться о безопасности от заражения вирусами, распространяющихся через макросы программ Word, Excel и другие приложения Windows.

В Excel имеется три степени защиты от вирусов: низкая, средняя и высокая. Низкая степень защиты не обеспечивает защиту от заражения программы вирусом. Этот режим рекомендуется использовать в том случае, если в компьютере установлена антивирусная программа и есть уверенность, что заражение вирусами исключено. При средней степени защиты решение о запуске потенциально опасных вирусов принимается пользователем. При высокой степени защиты разрешается запуск только подписанных макросов из надежных источников.

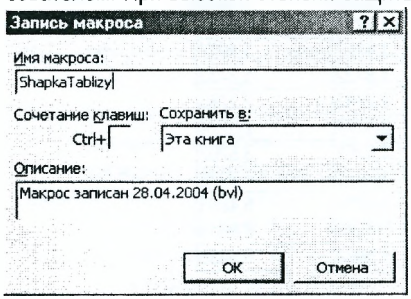


Рис. 6.21. Запись макроса

Для работы в сети рекомендуется устанавливать среднюю степень защиты. Для этого необходимо выполнить команду ***Сервис, Макрос, Безопасность*** и установить переключатель для средней степени защиты.

Предупреждение. Если макрос не запускается, проверьте установленную степень защиты (среднюю).

Назначение макроса кнопке панели инструментов

Для использования созданного макроса необходимо ввести команду ***Сервис, Макросы*** выбрать в списке нужный макрос и дать команду ***Выполнить***. Можно использовать для запуска макроса комбинацию клавиш, а наиболее удобно использовать макрос, если он назначен кнопке панели инструментов или кнопке, установленной на рабочий лист.

Для присвоения макроса кнопке меню выполните следующие операции:

введите команду ***Сервис, Настройка***, выберите закладку ***Команды*** окна диалога Настройка и в списке "Категории" выделите ***Макросы***. Выберите в списке "Команды" "Настраиваемую кнопку" и перетащите ее на панель инструментов. Вызовите контекст-

ное меню кнопки и выберите в нем команду ***Назначить макрос***. Выберите в списке макросов нужный макрос.

Для удаления кнопки введите команду ***Сервис, Настройка***, откройте закладку ***Команды***, выберите категорию Макросы и перетащите кнопку с панели инструментов в окно диалога.

6.8.2. СОЗДАНИЕ ФУНКЦИЙ ПОЛЬЗОВАТЕЛЯ С ПОМОЩЬЮ VBA

В приложение Excel интегрирована неполная версия языка программирования Visual Basic под названием Visual Basic, Application Edition или Visual Basic for Applications, которая обладает частью стандартных функциональных возможностей. Кроме того, эта специальная версия поддерживает объекты, которые позволяют обращаться к содержимому ячеек и управлять приложением Excel.

Создание процедур и функций в Visual Basic

Процедура или функция представляют собой фрагмент кода, выполняемый как один блок, и имеет обязательно заголовок и завершающую инструкцию, между которыми и находится собственно выполняемый код. Синтаксис заголовка процедуры:

```
Sub Имя_процедуры(аргументы)
    Тело процедуры
End Sub
Function Имя_процедуры(аргументы)
    Тело процедуры
End Function
```

Для создания функции пользователя необходимо ввести команду ***Сервис, Макрос, Редактор Visual Basic***. В редакторе VB выбрать команду ***Вставка, Модуль***, а затем ***Вставка, Процедура***. Написать программный код и сохранить программу командой ***Файл (File), Сохранить*** и выйти из Excel (***Close and Return to Microsoft Excel***) (см. также раздел 5.6).

Функции пользователя используются так же, как и встроенные функции Excel. Для ввода функций пользователя можно воспользоваться Мастером функций, категория ***" Определенные пользователем"***.

Пример 6.10. Создать функцию пользователя для табулирования функции одной переменной $y=x^2$ (Листинг 6.21).

Листинг 6.21. Табулирование функции одной переменной

	A	B	C	D	E	F
1	*Исходные данные*			**Текст программы**		
2	N=	7		Public Function Tab1perem(n As Byte, _		
3	X=	1		x As Single, dx As Single) As Variant		
4	Dx=	0,5		Dim i As Byte		
5	*Аргумент*	*Функция*		Dim a(50,1) As Single		
6	1	1		For i = 0 To n - 1		
7	1,5	2,25		a(I, 0) = x: a(i, 1) = x * x		
8	2	4		x = x + dx		
9	2,5	6,25		Next I		
10	3	9		Tab1perem = a		
11	3,5	12,25		End Function		
12	4	16				

Порядок выполнения:

- введите команду ***Сервис, Макрос, Редактор Visual Basic***;
- в редакторе VB выберите команду ***Вставка, Модуль***, а затем ***Вставка, Процедура***;
- установите в окне диалога переключатели ***Function*** и ***Public***, запишите в строке ввода Name имя функции: *Tablperem;*
- напишите текст программы;
- сохраните программу командой ***File, Save*** и вернитесь в программу Excel командой ***Close and Return to Microsoft Excel (Файл, Сохранить и выйти из Excel)***.

Из данного примера видно также, что в функциях можно использовать массивы. Массивы могут быть объявлены как статические (см. листинг 6.21), так и как динамические. Для объявления динамического массива объявите его сначала без указания размерности в разделе General (Общие), а затем в теле функции с указанием размерности с помощью оператора ReDim, например:

```
Dim A() As Single 'объявление массива в разделе Общие
---------------------------------------------
Public Function Tab2perem(n As Byte, m As Byte, _
x As Single, dx As Single) As Variant
        ReDim A(n,m) As Single  ' объявление массива в теле функции
        ...
End Function
```

Применение функции пользователя

- внесите в таблицу исходные данные N, X, Dx, как показано на листинге 6.21;
- выделите область ячеек A6:B12;
- вызовите мастер функций командой Вставка, Функция или щелчком мыши по соответствующей пиктограмме в панели инструментов Стандартная; - выберите в списке "Категории" окна мастера функций категорию "***Определенные пользователем***", а в списке "Функция" выберите имя функции *Tablperem*;
- внесите адреса ячеек данных в окне диалога и нажмите комбинацию клавиш Ctrl+Alt+Enter.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите порядок создания макроса.
2. Как назначить макрос кнопке панели инструментов?
3. Как назначить макросу комбинацию клавиш?
4. Как изменить надпись и значок на кнопке панели инструментов, предназначенной для управления макросом?
5. Как создать функцию пользователя?
6. Как используется функция пользователя?
7. Как установить/изменить защиту от макровирусов?

ЗАКЛЮЧЕНИЕ

Макросы являются удобным средством автоматизации типовых операций в электронной таблице. Другим способом повышения эффективности работы является создание процедур и функций пользователя. При создании функций пользователя они автоматически помещаются в список функций категории "Пользовательские" и используются так же как и встроенные функции.

Для защиты от макровирусов следует обязательно устанавливать защиту. Рекомендуется устанавливать среднюю степень безопасности.

6.9. ОСНОВЫ СОЗДАНИЯ И ИСПОЛЬЗОВАНИЯ СПИСКОВ В ЭЛЕКТРОННОЙ ТАБЛИЦЕ

6.9.1. ОБЩИЕ СВЕДЕНИЯ, ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Списки или базы данных в электронной таблице Excel представляют собой подобие информационно-справочных систем, известных как системы управления базами данных (СУБД).

СУБД - один из классов программных средств, предназначенных для создания, ведения и использования баз данных, справочных, информационно-поисковых систем. Основными компонентами информационной системы являются: база данных, система управления базами данных, прикладная программа и интерфейс. База данных содержит интересующую пользователя информацию, а также описание структуры хранимых данных. СУБД выполняет типовые процедуры управления данными, осуществляет взаимодействие с прикладной программой. Прикладная программа реализует требуемый алгоритм ведения диалога пользователя с информационной системой, ввода и контроля запросов, организации информационного поиска, выборки и представления данных в виде справок и отчетов. Взаимодействие между прикладной программой и СУБД осуществляется с помощью специальных операторов или команд языка управления базой данных.

Возможны три модели баз данных: *сетевые, иерархические и реляционные*. Сетевые и иерархические СУБД получили наибольшее распространение на больших- и мини-ЭВМ. На ПК используется преимущественно реляционная модель данных. Сетевые СУБД используют модель представления данных в виде произвольного графа. В иерархических СУБД данные представляются в виде древовидной структуры. Реляционная модель ориентирована на представление данных в виде таблицы. В Excel реализована реляционная база данных.

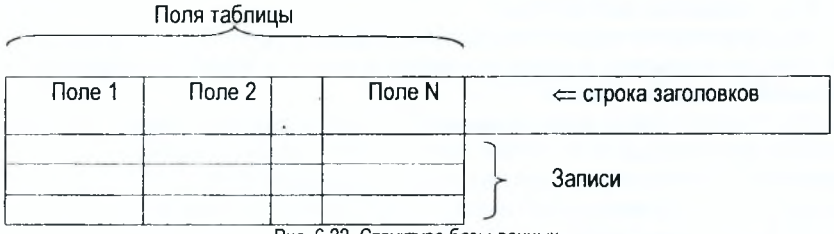


Рис. 6.22. Структура базы данных

Таблица реляционной БД (рис.6.22) представляет собой двухмерный массив и обладает следующими свойствами: каждый элемент таблицы - это один элемент данных, повторяющиеся группы отсутствуют; все столбцы (колонки) в таблице однородные. Это означает, что все элементы одного столбца имеют одинаковую природу, например: марка автомобиля или размер заработной платы. Столбцам присвоены уникальные имена; в таблице нет одинаковых строк; в операциях с таблицей ее строки и столбцы могут просматриваться в любом порядке и в любой последовательности безотносительно к их информационному содержанию и смыслу.

Каждая строка таблицы – это запись, каждая колонка таблицы – поле записи. Размещение в одной строке таблицы определенных элементов данных означает установление между ними связи или отношения. Например, если база данных содержит сведения о запасных частях к автомобилям, то в одной строке могут быть помещены сведе-

ния о запасных частях к автомобилю конкретной марки. То есть, данные в одной строке связаны между собой тем, что принадлежат одной марке автомобиля. Строка таблицы с этими данными представляет собой один конкретный экземпляр отношения данного типа или его *кортеж*, а всю таблицу в целом называют *отношением*. Таким образом, при описании реляционной модели данных отношением называют всю таблицу в целом как совокупность конкретных экземпляров отношения. Слова "отношения" и "реляционный" (от латинского relation - отношение) представляют собой синонимы.

Совокупность значений элементов данных, размещенных в одном столбце таблицы и определяющих некоторую характеристику или свойство объектов, описываемых строками таблицы, называют атрибутом отношения. Количество элементов данных в кортеже (количество столбцов в таблице) определяет степень отношения. Если таблица включает n столбцов, то она представляет собой отношение степени n. Количество кортежей в отношении (число строк в таблице) определяют его мощность - m. Тогда общее количество элементов данных в отношении степени n будет равно n x m.

Атрибут, значение которого идентифицирует кортеж, то есть позволяет однозначно выделить его из других кортежей данного отношения, называется ***ключевым атрибутом*** или просто ***ключом***. Ключ может включать несколько атрибутов - составной ключ или представлять собой только часть значения атрибута - частичный ключ. В приведенном выше примере в качестве ключа может быть марка автомобиля, что позволяет однозначно выделить кортеж из всего отношения.

Программа Excel позволяет импортировать и обрабатывать данные из других баз данных. Excel позволяет создавать собственные однотабличные базы данных, устанавливать между ними связи с помощью функций ПРОСМОТР, ВПР, ГПР, а также OLE-автоматизации, осуществлять сортировку, поиск и извлечение данных.

6.9.2. СОЗДАНИЕ БАЗ ДАННЫХ

Базу данных рекомендуется создавать на отдельном листе. В этом случае программа быстрее использует команды сортировки и фильтр, а также исключается возможность испортить другие данные.

Для создания списка может использоваться любой диапазон ячеек. Тогда каждый столбец диапазона считается полем, которое может содержать строку длиной до 255 символов. Соответственно каждая строка диапазона будет считаться записью. Для создания и работы со списками Excel имеет специальные команды, функции и методы.

Листинг 6.22. Пример базы данных

Дата	Откуда	Вид	Количество	Объем	Цена	Стоимость
Сентябрь	Братск	Бумага	22500	45	3500	78750000
Сентябрь	Братск	Ватман	15600	31	2400	37440000
Сентябрь	Вологда	Цемент	13600	27	5800	78880000
Сентябрь	Тюмень	Клей столярный	11000	22	1200	13200000
Октябрь	Братск	Картон	12000	48	5600	67200000
Октябрь	Вологда	Плитка облицовочная	13500	27	3200	43200000

Щелкните правой клавишей мыши по ярлычку номера страницы и введите название базы данных (по ее содержанию).

В первую строку диапазона, отведенного для создания списка, записываются имена полей. Имена полей должны быть, по возможности, простыми, краткими и описательными, при этом они не могут занимать более одной строки таблицы. В последующие строки записываются данные. Первая запись не должна ничем отделяться от строки заголовков, в списке не должно быть одинаковых записей, пустых строк и колонок. Пример базы данных приведен на листинге 6.22.

Для работы со списками используются команды ***Сортировка, Фильтр и Форма***, которые находятся в пункте главного меню ***Данные.***

Команда ***Форма*** используется для создания списка, добавления данных в список, просмотра списка и поиска данных по заданному критерию. При вводе команды открывается окно диалога (рис. 6.23). Оно позволяет просматривать базу данных, перемещаясь по ней с помощью команд *Далее* и *Назад*, добавлять, удалять и редактировать зна-

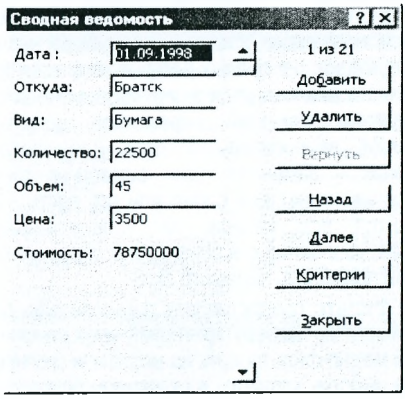


Рис. 6.23. Окно диалога команды Форма

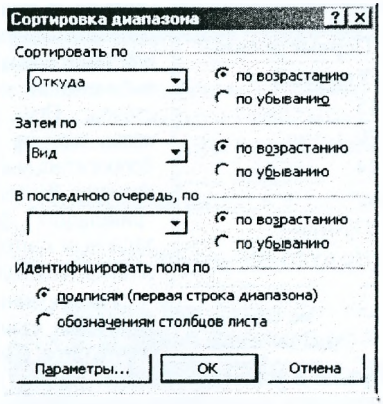


Рис.6.24. Окно диалога команды Сортировка

чения полей базы данных. Добавляемые записи помещаются в конец списка. При этом вычисляемые поля (Стоимость) недоступны для редактирования. Команда *Критерий* позволяет осуществлять поиск записей по заданному критерию:

- щелкните мышью по кнопке Критерий – открывается пустая форма;
- введите в нужное поле значение критерия.

Например, если требуется найти товары поступившие из Омска, тогда укажите в поле Откуда "Омск" – на экран будет выведена первая запись, удовлетворяющая условиям поиска. Просмотрите другие записи, используя кнопки Далее и Назад.

Вид	оличество	Объем
Бумага	22500	45
Ватман	15600	31
Цемент	13600	27
Клей столярный	11000	22
Картон	12000	48

Рис. 6.25. База данных, режим Автофильтра

Команда ***Сортировка*** позволяет отсортировать выделенный диапазон по значениям одного, двух или трех полей (рис. 6.24). Принцип сортировки аналогичен сортировке в Microsoft Word. Сортировку можно проводить

по возрастанию или убыванию значения соответствующего поля. В случае сортировки базы данных столбцы будут фигурировать под названиями полей, а в случае сортировки простого списка – под названиями столбцов. В группе "Идентифицировать поля по" два переключателя. Если активизировать переключатель "подписями (первая строка диапазона)", то в списках будут указаны поля базы данных, а если активизировать переключатель "обозначениями столбца листа", то в списки будут выведены заголовки столбцов таблицы. Кнопка "Параметры" позволяет указать дополнительные условия сортировки: учитывать ли регистр, сортировать строки диапазона или столбцы диапазона.

Команда ***Фильтр***. Команда Фильтр содержит опции *Автофильтр*, *Отобразить все* и *Расширенный фильтр*.

При выборе опции ***Автофильтр*** на каждом поле появляется кнопка раскрывающегося списка (Рис.6.25.). Если щелкнуть мышью по кнопке раскрывающегося списка, то открывается список параметров (Рис.6.26). Выбор требуемого параметра осуществляется щелчком мыши. Параметр ***Условие*** выводит на экран диалоговое окно ***Пользовательский автофильтр***, которое позволяет объединить два параметра по логическому условию И или ИЛИ (Рис.6.27.). В левых раскрывающихся списках выбираются логические условия для выбора числовой или текстовой информации, а в правых раскрывающихся списках выби-раются из списка или вводятся значения параметров для отбора. При указании значений параметров допускается использование маски: вопросительный знак или звездочка. Вопросительный знак – заменяет один символ в текущей позиции. Звездочка за-меняет все слово или его часть. После применения фильтра все записи, не удовлетво-ряющие заданным критериям, убираются с экрана. Для отображения всех записей необходимо применить параметр ***Все***.

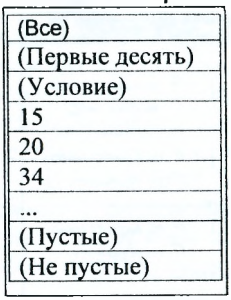


Рис. 6.26.
Открытый список

Расширенный фильтр предоставляет пользователю дополнительные возможности по выбору критериев и формированию результатов: список можно фильтровать на месте или скопировать результат в указанный диапазон; условия для выбора задаются в отдельном диапазоне рабочего листа; можно использовать при выборе только уникальные записи.

Блок критериев для расширенного фильтра содержит условия для поиска и выборки данных. Он может располагаться в любом месте электронной таблицы. Блок критериев состоит из двух или более строк. В первой строке задаются имена полей, а в последующих строках – значения критериев поиска. Критерием может быть текстовая или числовая константа, логическая функция или логическое выражение. Если критерий содержит несколько строк, то считается, что эти строки связаны функцией ИЛИ. Если строка критерия содержит несколько полей, то считается, что эти поля связаны функцией И. Примеры блоков критериев для расширенного фильтра приведены на листинге 6.23.

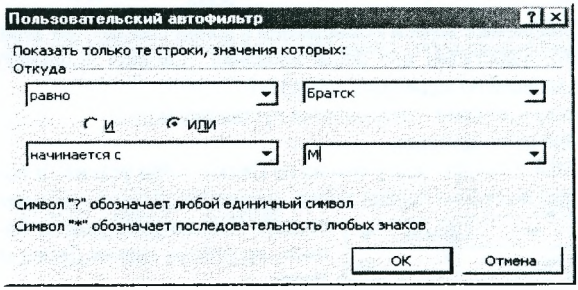


Рис. 6.27. Создание пользовательского фильтра

Листинг 6.23. Примеры блоков критериев расширенного фильтра

	E	F	G	H	I
1	**Наименование**		**Наименование**	**Дата_поступления**	**Цена**
2	Телевизоры		Телевизор	01.01.2000	<200000
3	Холодильник				
4	Пылесос		**Наименование**	**Дата_поступления**	**Цена**
5			Телевизор		
6				01.01.2000	
7					<200000

В блоке критериев E1:E4 три наименования объединены по схеме ИЛИ. То есть при наличии на складе указанных товаров все они будут включены в выходной список. В блоке критериев G1:I2 три параметра объединены по схеме И. Это значит, что для выбора товара должны быть выполнены все три условия: в базе данных будет отыскиваться телевизор, поступивший на склад первого января 2000 года по цене меньше 200000 рублей. В блоке критериев G4:I7 три разных критерия объединены по схеме ИЛИ, то есть будут отыскиваться телевизор, все товары, поступившие первого января 2000 года, и все товары по цене меньше 200000 рулей.

В поле критерия для текстовых данных могут использоваться шаблоны ?, *. Символ "?" заменяет один знак в указанной позиции. Например, "ию?ь" совпадает с "июнь" и "июль". Символ * заменяет все слово или его часть. Например, "авто*" соответствует "автомобиль", "автокар" и т. д.

Блок критериев для расширенного фильтра рекомендуется задавать выше базы данных (списка), отделяя его от базы данных одной строкой.

Для получения выборки с помощью расширенного фильтра необходимо:

- создать блок критериев;

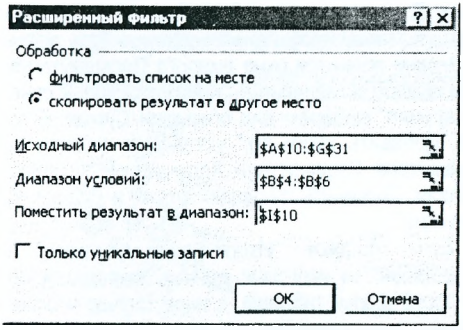


Рис. 6.28. Создание расширенного фильтра

- выделить базу данных (для выделения базы данных достаточно установить курсор в любую ячейку этой базы данных);
- ввести команду Данные, Фильтр, Расширенный фильтр;
- указать в окне диалога Расширенный фильтр (рис.6.28) исходный диапазон, диапазон условий;
- если предполагается получить выборку на месте, то следует нажать клавишу ОК. Если предполагается поместить выборку в другое место, то следует активизировать флажок "Скопировать результат в другое место" и указать в строке ввода "Поместить результаты в диапазон" начальную ячейку выходного блока данных.

При обработке баз данных полезными являются функции обработки данных:

ДСРЗНАЧ() – среднее значение элементов базы данных, соответствующих заданному критерию;

БСЧЕТ() - количество записей в базе данных, удовлетворяющих заданному критерию;

ДМАКС() – максимальное значение записей, соответствующих критерию, заданному в поле;

ДМИН() - минимальное значение записей, соответствующих критерию, заданному в поле;
БДСУМ() – сумма значений записей, соответствующих критерию, заданному в поле и другие функции.

Все функции базы данных имеют одинаковый синтаксис:

<Имя_функции>(*база_данных, поле, критерий*)

С помощью аргумента *база данных* в функцию передается диапазон ячеек, подлежащих обработке. Можно ссылаться на имя Базы данных.

Аргумент *поле* идентифицирует поле базы данных, с которым предполагается проводить вычисления. Для обозначения поля можно использовать как номер столбца в базе данных, так и имя поля базы данных.

Аргумент *критерий* соответствует ссылке на диапазон условий. Диапазон условий задается так же как и при формировании блока критериев для расширенного фильтра.

Например, для вычисления суммы количества бумаги, поступившей от поставщиков (рис.6.26), следует ввести в ячейку формулу:

БДСУММ(A10:G31;D10;C4:C5)

или

БДСУММ(ПокупныеИзделия;"Количество";C4:C5)

Здесь ПокупныеИзделия – имя базы данных, "Количество" – имя поля, C4:C5 – блок критериев.

6.9.3. АНАЛИЗ ДАННЫХ

Для анализа данных можно использовать команды ***Итоги*** и ***Сводные таблицы*** меню Данные.

Итоги

Команда Итоги позволяет получать сводные данные по числовым параметрам: сумму, минимум, максимум, среднее значение и другие статистические данные. Предварительно необходимо определить, по какому параметру требуется группировать итоги и отсортировать данные по этому параметру. В примере на рис. 6.29, Листинг 6.24, в качестве такого параметра выбран вид продукции. Затем необходимо выделить базу данных и ввести команду ***Данные***, ***Итоги***. На экране появится окно диалога Промежуточные итоги (рис. 6.29). Выберите в списке "При каждом изменении в:" параметр *Вид*, в списке "Операция" выберем вид операции *Сумма*. В списке "Добавить итоги по:" установим флажки для параметров Стоимость и Количество. Установим флажки "Заменить текущие итоги" и "Итоги под данными".

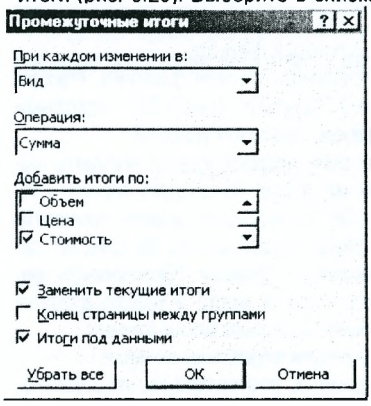


Рис. 6.29. Окно диалога Промежуточные Итоги

Если флажок "Итоги под данными" не установлен, то итоговые данные выводятся сразу же после шапки таблицы, в ином случае итоговые данные будут размещены в конце таблицы.

Флажок "Конец страницы между группами" позволяет разбить данные по видам и напечатать их на отдельных листах. При малом объеме данных это делать не целесообразно.

Кнопка Убрать все удаляет все итоги и переводит базу данных в исходное состояние.

Листинг 6.24. База данных после выполнения команды Итоги

	A	B	C	D	E	F	G
36	Дата	Откуда	Вид	Количество	Объем	Цена	Стоимость
37	Сентябрь 98	Братск	Бумага	22500	45	3500	78750000
38	Сентябрь 98	Мурманск	Бумага	42500	85	2500	106250000
39	Октябрь 98	Мурманск	Бумага	31000	62	2900	89900000
40			**Бумага Всего**	96000			274900000
41	Сентябрь 98	Братск	Ватман	15600	31	2400	37440000
42	Ноябрь 98	Мурманск	Ватман	12000	24	2000	24000000
43			**Ватман Всего**	27600			61440000
44	Ноябрь 98	Мурманск	Калька	14500	29	4300	62350000
45			**Калька Всего**	14500			62350000
46	Сентябрь 98	Братск	Картон	13600	27	5800	78880000
47	Октябрь 98	Братск	Картон	12000	48	5600	67200000
48	Сентябрь 98	Мурманск	Картон	32000	64	5200	166400000
49	Октябрь 98	Мурманск	Картон	26500	53	5090	134885000
50			**Картон Всего**	84100			447365000
51	Сентябрь 98	Братск	Клей	11000	22	1200	13200000
52			**Клей Всего**	11000			13200000
53			**Общий итог**	233200			859255000

После установки всех параметров щелкните по кнопке ОК - база данных преобразуется к виду, представленному на листинге 6.24. Под каждым видом продукции помещены промежуточные итоги, а внизу таблицы Общие итоги. Слева от таблицы появиляются кнопки для управления уровнем просмотра. При щелче по кнопке 2 на экране остаются только промежуточные и общие итоги, при щелчке по кнопке 1 – только общие итоги.

Сводные таблицы

Команда Сводные таблицы позволяет формировать новые таблицы на базе существующих таблиц в различных разрезах и проводить анализ полученных результатов.

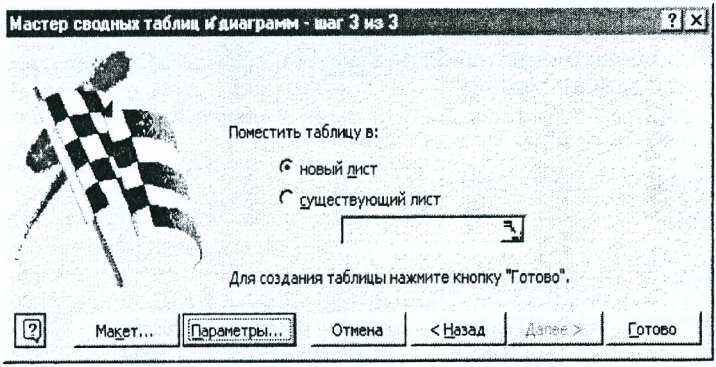


Рис. 6.30. Мастер сводных таблиц, шаг 3

Создание сводных таблиц осуществляется с помощью Мастера сводных таблиц за четыре шага.

На первом шаге предлагается выбрать источник данных и вид отчета. В качестве

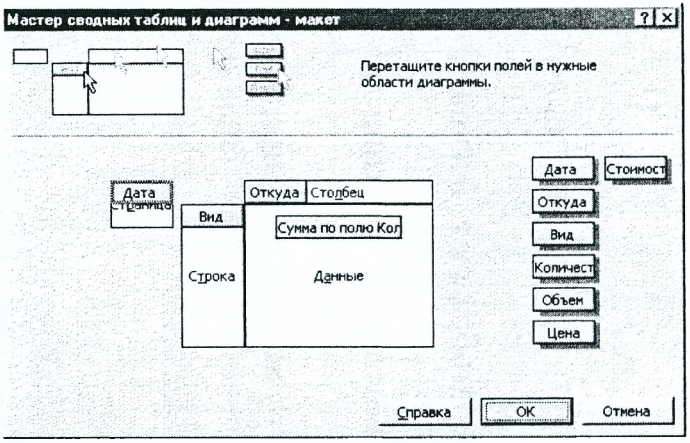


Рис. 6.31. Окно диалога создания макета сводной таблицы

источника данных может быть база данных Excel или внешняя база данных, например, база данных Microsoft Access, Visual FoxPro, dBase-Word.

На втором шаге предлагается указать диапазон, содержащий исходные данные.

На третьем шаге можно указать место для сводной таблицы (на текущем листе или на новой странице), создать макет сводной таблицы и настроить некоторые параметры (рис. 6.30.). При выборе команды Макет открывается одноименное окно диалога (рис. 6.31). На макете обозначены четыре зоны: Страницы, Строка, Столбец, Данные. В эти зоны необходимо поместить кнопки требуемых полей базы данных, расположенных справа от макета. Заполнение зон Страницы, Строка, Столбец, Данные осуществляется претаскиванием кнопок базы данных мышью. Поле Страница выполняет роль фильтра и позволяет выполнить группировку данных по некоторому полю базы данных, например, по дате поступления, отправителю или виду. В зону Строки помещают данные, которые должны быть помещены в строки сводной та блицы. А в зону Столбец помещают имена полей, которые должны размещаться в столбцах базы данных. В зону Данные помещаются, как правило, числовые поля, но можно помещать и текстовые поля, тогда для обработки их используется функция Count.

Листинг 6.25. Сводная таблица

Дата	(Все)				
Сумма по полю Количество	Откуда				
Вид	Братск	Вологда	Мурманск	Тюмень	Общий итог
Бумага	22500	20800	73500	30000	146800
Ватман	15600	10000	12000	7500	45100
Калька		7500	14500	6500	28500
Картон	25600		58500	53000	137100
Клей	11000				11000
Общий итог	74700	38300	158500	97000	368500

Рассмотрим для примера фрагмент базы данных, приведенный на листинге 6.22. Поместим в зону Страница кнопку *Дата*, в зону Строка – кнопку *Вид*, в зону Столбец –

кнопку *Откуда* и в зону Данные – кнопку *Количество* (рис. 6.31). При установке в зону Данные кнопки Количество по умолчанию предлагается выполнить суммирование значений данных по соответствующему полю и соответственно имя кнопки изменяется. Можно выбрать другой вид операции. Для этого щелкните дважды мышкой по кнопке *Сумма по полю Количество*. Для завершения операции щелкните по кнопке ОК – программа возвращается в окно шага 3 (рис.6.30). Для завершения формирования сводной таблицы щелкните по кнопке Готово. Результат приведен на Листинге 6.25. В списке Все содержится список дат поступления товаров. Список Откуда содержит список источников поступления. Эти списки снабжены флажками, которые позволяют управлять выводом информации на экран. Например, открыв список Откуда, можно снять флажки у Вологды, Мурманска, Тюмени. Тогда на экране останутся данные только по Братску. В списке дата по умолчанию флажки установлены у всех элементов списка. Можно установить флажок только для одной, интересующей Вас даты, тогда на экране будет выведен список товаров, поступивших в указанную дату.

6.9.4. КОНСОЛИДАЦИЯ

Еще одной операцией с базами данных является ***консолидация***. Консолидация – означает совместную обработку данных двух и более диапазонов, например, требуется сложить данные из столбцов двух диапазонов. При этом диапазоны могут находиться в одной таблице, на одном и том же листе или на нескольких листах рабочей книги и даже в разных книгах. Консолидация может проводиться двумя способами:

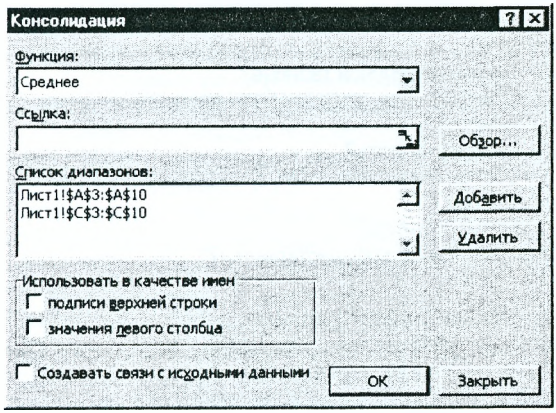


Рис. 6.32. Окно диалога Консолидация

1. Консолидация по расположению ячеек – в этом слу-чае состав и порядок расположения данных во всех диапазонах одинаков;

2. Консолидация по категориям – в этом случае консолидация проводится на основании одинаковых подписей строк и столбцов диапазонов.

Введем команду ***Данные, Консолидация*** – на экране появится окно диалога Консолидация (рис. 6.32). Список "Функция" позволяет выбрать операцию над консолидируемыми данными. При консолидации доступны все функции статистических итогов (сумма, минимум, максимум и. д.). Строка ввода "Ссылка" позволяет указать или выбрать диапазон консолидируемых данных. "Список диапазонов" хранит списки диапазонов консолидируемых данных. Кнопка Добавить позволяет добавлять данные в список диапазонов через строку "ссылка".

Флажки в группе "Использовать в качестве имен" используются при втором способе консолидации. При установке флажка "Создавать связи с исходными данными" в случае изменения исходных данных одновременно будет изменяться и результат. Связи нельзя использовать, если исходная область и область назначения находятся на одном листе. После установки связей нельзя добавлять новые исходные области и изменять исходные области, уже входящие в консолидацию.

Кнопка Обзор служит для выбора данных в других книгах.

Пример 6.9. Пусть требуется найти среднее значение для каждой строки данных в столбцах А и В. Результаты поместить в столбец С.

Листинг 6. 26

A	B	C
1	2	1,5
2	3	2,5
3	4	3,5
4	5	4,5
5	6	5,5
6	7	6,5
7	8	7,5
8	9	8,5

Алгоритм работы:

- выделите диапазон ячеек, равный высоте столбца А, без учета заголовков;
- введите команду Данные, Консолидация;
- активизируйте строку ввода "Ссылка", выделите первый диапазон в столбце А и щелкните по кнопке Добавить;
- активизируйте строку ввода "Ссылка", выделите второй диапазон в столбце В и щелкните по кнопке Добавить;
- выберите функцию "Среднее" в списке Функция;
- щелкните по кнопке ОК.

Состояние окна диалога Консолидация после ввода данных и выбора функции приведено на рис. 6.32, а результат – на листинге 6.26.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1.Что такое база данных?
2. Какие типы баз данных существуют?
3. Какой тип базы данных реализован в электронной таблице?
4. Какие требования необходимо соблюдать при создании базы данных?
5. Что такое автофильтр, как он создается?
6. Что такое расширенный фильтр, как его создать?
7. Как создать блок критериев а) простой, б) по схеме ИЛИ, в) по схеме И?
8. Для чего предназначена команда Итоги?
9. Что такое сводные таблицы, как они создаются?
10. Для чего необходима консолидация, как она осуществляется?

ЗАКЛЮЧЕНИЕ

Электронная таблица Excel позволяет пользователю создавать и вести базы данных. При этом под ведением базы данных подразумевается редактирование базы данных, добавление и удаление записей, поиск и извлечение данных по заданному критерию.

Каждую базу данных рекомендуется создавать на отдельном листе.

Добавление, удаление записей и поиск информации по заданному критерию осуществляется с помощью команды Дата, Форма.

Выборка и извлечение данных осуществляется с помощью фильтров: автофильтр и расширенный фильтр. Автофильтр позволяет создавать критерии для выборки данных по соответствующему полю по схеме И и ИЛИ на месте базы данных. Расширенный фильтр позволяет осуществлять выборку и извлечение данных как на месте базы данных, так и в другое место в пределах рабочего листа. Для использования расширенного фильтра необходимо создавать отдельный блок критериев.

Для подведения промежуточных итогов используется команда Итоги. Перед использованием команды база данных должна быть отсортирована по тому полю, которое будет использоваться для группировки данных.

Команда Сводные таблицы позволяет создавать на основе существующей базы данных новые базы данных, упорядоченные по строкам и полям. Сводные таблицы используются для анализа информации, содержащейся в базах данных.

При необходимости использования и совместной обработки данных из разных таблиц или диапазонов одной таблицы можно воспользоваться командой Консолидация.

7. ВВЕДЕНИЕ В ИНТЕРНЕТ

7.1. ОБЩИЕ СВЕДЕНИЯ ОБ ИНТЕРНЕТ

Интернет – это глобальная сеть компьютерных сетей, соединенных каналами связи и передающих информацию друг другу по определенным правилам, называемых протоколами. В качестве каналов связи могут использоваться обычные телефонные линии. В этом случае для подключения компьютера к сети используется устройство, называемое модемом (от комбинации слов модуляция/демодуляция). Предоставляют доступ к сети Интернет фирмы или организации, называемые провайдерами (от английского provide - обеспечивать). Провайдер – это организация или лицо, реализующее услуги Интернет в данном регионе.

Рождение Интернет относят к 60 – 70 годам. Первоначально Министерство Обороны США создало сеть, которая называлась ARPAnet. Эта сеть создавалась в военно-промышленной сфере для поддержки научных исследований, - в частности, для исследования методов построения сетей, устойчивых к частичным повреждениям, получаемым, например, при бомбардировке авиацией и способных в таких условиях продолжать нормальное функционирование. В 80-х годах к Интернету подключились образовательные учреждения, государственные организации, американские и иностранные фирмы различного направления деятельности.

World Wide Web (WWW) - наиболее новая и быстро развивающаяся сегодня служба Internet. Она имеет почти неограниченный потенциал в плане сбора, распространения и изучения информации. Обеспечиваемые ей графические межплатформенные средства завоевывают все большую популярность у пользователей и компаний, которым необходимо собирать информацию, обмениваться информацией и предлагать коммерческую информацию в Internet.

Для работы в Интернет используется протокол передачи гипертекстовых сообщений HTTP - HyperText Transfer Protocol (протокол передачи гипертекста). Гипертекстовые документы (Web-страницы) создаются с помощью языка разметки гипертекста HTML – HyperText Markup Language. Такие документы могут содержать графику и гипертекстовые ссылки. При щелчке мышью гипертекстовая ссылка выводит пользователю другой документ. Таким образом, эта ссылка содержит "указатель" на документ, который становится доступным при нажатии кнопки мыши. Web-страницы размещаются на отдельных компьютерах, называемых Web-серверами и принадлежащих отдельным организациям или частным лицам. Сервер - это основополагающее понятие Интернета. Под этим термином понимают удаленный компьютер, на котором работает серверная программа, выполняющая обработку запросов пользователей: идентификацию пользователей, проверку их полномочий, прием данных от пользователей и передачу им данных. Группу Web-страниц, объединенных по смыслу и имеющих одинаковое дизайнерское решение, называют Web-сайтом.

Интернет обладает большим набором сервисных функций, значительно увеличивающих к нему интерес различных категорий пользователей. Среди них следует выделить следующие: электронная конференция, чат, электронная доска объявлений, форум, e-mail – электронная почта и т.д.

Чат - одновременный разговор нескольких абонентов через сеть. Обычно это диалог в текстовом режиме. Диалог возможен как при использовании броузера, так и через специального клиента.

Форум - это инструмент для общения на сайте. Сообщения в форуме в чем-то похожи на почтовые сообщения: каждое из них имеет автора, тему и собственно содержание. Но для того, чтобы отправить сообщение в форум, не нужна никакая дополнительная программа - нужно просто заполнить соответствующую форму на сайте.

Доска обьявлений - (англ. www-board, синонимы: веб-доска, веб-борд) Сайт, где вы можете разместить свое объявление, аналогичен доске объявлений или газете бесплатных объявлений в обычной жизни. Как правило, такие доски являются бесплатными, тематически организованными. На такой доске можно легко разместить свое объявление, указав его тематику и срок хранения, и оно появится на доске практически сразу после отправки.

Электронная почта – это:

- способ пересылки сообщений в Интернете;
- электронный адрес человека или организации, на который поступает корреспонденция.

Электронная коммерция - (англ. e-commerce, синоним - e-коммерция) в самом широком смысле - коммерческая деятельность с использованием Интернета. Как правило, имеются в виду механизмы, позволяющие упростить работу и продавцов и покупателей.

7.2. ОРГАНИЗАЦИЯ РАБОТЫ В ИНТЕРНЕТ

7.2.1. АДРЕСАЦИЯ В ИНТЕРНЕТ

Для обмена данными между абонентами использует набор правил, который получил название протокол. Набор протоколов, используемых в Интернет, существует под общим названием TCP/IP. Все протоколы сети Интернет разбиты на четыре уровня:

- 1-й уровень – Прикладной, служит для организации взаимодействия с пользователем. К нему относятся следующие протоколы: HTTP – доступ к удаленным гипертекстовым базам данных во всемирной паутине, Gopher – доступ к ресурсам всемирного пространства, WAIS – распределенная информационно-поисковая система, FTP – копирование файлов, Telnet – эмуляция терминала и другие.
- 2-й уровень - Протокол управления передачей TCP;
- 3-й уровень - Протокол межсетевого взаимодействия IP;
- 4-й уровень - уровень доступа к сети, не регламентируется. Здесь могут использоваться протоколы Ethernet, Token Ring, FDDI, X.25, SLIP, PPP

Каждый компьютер в Интернете имеет свой уникальный номер, так называемый IP – адрес. Это адрес, которым руководствуются компьютеры общаясь между собой в Интернет. Существует две разновидности адресов: *адреса компьютеров* (узлов Интернет); *адреса информационных ресурсов*, расположенных на этих компьютерах. IP – адрес содержит четыре группы чисел (каждое в пределах от 0 до 255), разделенных точками. Например, 212.193.2.201 – адрес сервера Санкт-Петербургского отделения института "Открытое общество". Адреса назначаются организацией InterNIC, которая передает их провайдерам. Провайдеры в свою очередь распределяют адреса среди компьютеров своего участка сети.

Для удобства IP-адресам поставлены в соответствие символьные или доменные адреса. Домен – это множество компьютеров, имеющих общую часть имени. Составные части доменного адреса называются сегментами: <сегмент> . <сегмент> . ... . <сегмент>. Сегменты образуют иерархическую систему. Самый правый сегмент является доменом верхнего уровня. Домены обычно различаются на географические и тематические. Географические доменные имена верхнего уровня – двухбуквенные. Они определяют принадлежность владельца имени к сети конкретной страны (например, ***by*** – Беларусь , ***ru*** - Россия, ***fr*** – Франция, ***us*** – Соединенные Штаты). Тематические адреса дают возмож-

ность представить сферы деятельности их владельцев: ***com*** – коммерческие фирмы (например, http://www.microsoft.com/); ***edu*** – образовательные учреждения; ***gov*** – государственные, ***mil*** – военные, ***org*** – прочие организации.). В доменном имени крайняя правая часть обозначает домен верхнего уровня, крайняя левая - имя собственно компьютера, остальные, справа налево, - набор вложенных друг в друга поддоменов, каждый следующий является частью предыдущего. Например, *sky.inp.nsk.ru* – Россия (*ru*), Новосибирск (*nsk*), Институт ядерной физики (*inp*, Institute for Nuclear Physics), компьютер *sky*. Всего в Интернет может быть использовано 4294967296 возможных IP-адресов.

Протоколы транспортного уровня управляют передачей данных из одной программы в другую. К протоколам транспортного уровня принадлежат TCP и UDP (User Datagram Protocol).. UPD проще TCP, потому что этот протокол не заботится о пропавших пакетах, расположении данных в правильном порядке и других тонкостях. UDP используется для тех программ, которые посылают только короткие сообщения и могут повторить передачу данных, если ответ задерживается.

Информационным ресурсам Интернет ставят в соответствие ***URL-адреса*** (от англ. Uniform Resource Locator – универсальный адрес ресурса). URL-адрес состоит из двух частей, разделяемых двоеточием. Первая (левая) часть указывает на то, к какому протоколу принадлежит ресурс и как получить к нему доступ, т.е. определяет конкретный сетевой протокол. Вторая часть URL-адреса сообщает, где расположен искомый ресурс. Таким образом, ***URL*** содержит информацию не только о том, где данный ресурс расположен, но и как к нему следует обращаться. Пример URL-адреса:

http://www.icape.nw.ru/.spb.osi.ru .

Указывать префикс http:// не обязательно. Броузер сам добавит эти символы, подразумевая, что пользователя интересуют скорее всего гипертекстовые ресурсы и, стало быть, к ним нужно обращаться по протоколу HTTP – оптимизированному для работы с

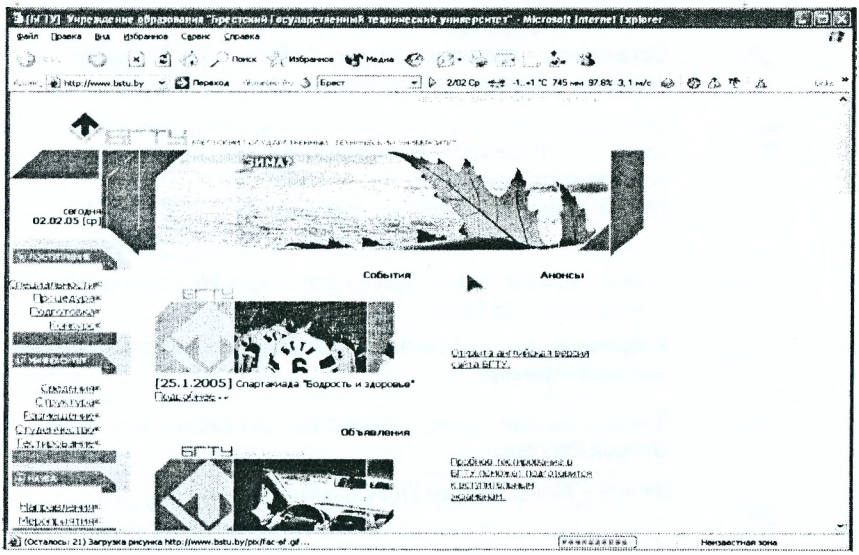


Рис. 7.1. Интерфейс броузера Internet Explorer

гипертекстовыми ресурсами Интернет. Персональный компьютер, рабочая станция, сервер или иное устройство, подключенное к сети и обеспечивающее пользователю связь с другими компьютерами, подключенными к Интернет (и, соответственно, имеющими IP-адрес), называется *хостом (host).*

7.2.2. СТРУКТУРА БРОУЗЕРА INTERNET EXPLORER И НАЗНАЧЕНИЕ ОСНОВНЫХ ЕГО ЭЛЕМЕНТОВ

Броузер – программное средство, обеспечивающее запрос, получение и предоставление документов в сети Интернет. Среди броузеров можно выделить: Internet Explorer, Opera, Netsscape Navigator. Наиболее распространенным броузером в настоящее время является Internet Explorer, интерфейс которого приведен на рис. 7.1.

Рассмотрим основные элементы броузера.

1) заголовок

В заголовке программы отображается адрес или название сайта, который открыт в данный момент. В правой части заголовка находятся три кнопки управления окном: свернуть; восстановить; закрыть.

2) кнопки навигации:

Кнопки навигации используются для более быстрой работы с броузером. Назначение основных кнопок:

Назад – вернуться на последнюю из просмотренных страниц.

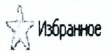
Вперед – возврат на страницу, которую просматривали до того, как нажали кнопку Назад.

Остановить – прерывает загрузку Web-страницы (например, если ожидание появления страницы занимает много времени).

Обновить – используется при получении сообщения о невозможности открыть какую-либо Web-страницу или для загрузки последнего варианта страницы.

Домой - возврат на стартовую страницу.

Поиск - выводит панель поиска для поиска Web-страниц по ключевому слову или фразе.



Избранное - выводит панель, отображающую закладки на избранные Web-страницы.

Почта – выводит меню с командами для работы в приложении **Outlook Express.**

Печать – вывод текущей Web-страницы на принтер.

3) меню

Меню содержит все команды, доступные для управления броузером.

4) строка адреса

Адресная строка служит для перехода на Web-страницу, по указанному адресу. В адресную строку нужно ввести URL-адрес, с клавиатуры или выбрать из раскрывающегося списка, например, http://www.iatp.unibel.by, а затем нажать кнопку ***Переход*** или клавишу ***ENTER*** на клавиатуре.

5) строка состояния

Строка состояния отображает состояние броузера. Существует несколько основных моментов:

- 50% - сайт загружен на 50 процентов;
- Готово - сайт загружен полностью.

Некоторые места в тексте Web-страницы подчеркнуты и выделены другим цветом на фоне остального текста, указатель мыши над ними принимает форму руки. Такой текст называется ***гиперссылкой*** (или просто ссылкой) и является указателем перехода на другие документы. После щелчка по ссылке, она изменит свой цвет, а потом содержимое окна исчезнет и будет загружена запрашиваемая страница. При повторной загрузке страницы все ранее использованные ссылки, как правило, выделяются особым цветом, что значительно экономит время и позволяет не просматривать одни и те же материалы несколько раз.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое Интернет?
2. Что такое HTTP и HTML?
3. Что понимается под Web – сайтом?
4. Что включает IP – адрес?
5. Что такое домен?
6. Что такое URL-адрес?
7. Что называется хостом?
8. Что понимается под броузером?
9. Какие броузеры получили наибольшее распространение?
10. Назовите основные кнопки интерфейса Internet Explorer и их назначение.

7.3. ПРИКЛАДНОЕ ИСПОЛЬЗОВАНИЕ СЕТИ ИНТЕРНЕТ

7.3.1. РАБОТА С ЭЛЕКТРОННОЙ ПОЧТОЙ

Пользователи сети Интернет имеют возможность обмениваться информацией друг с другом с помощью электронной почты – ***E-mail*** - одного из самых популярных и дешевых сервисов. Электронная почта осуществляет передачу сообщений между зарегистрированными адресами. Адрес электронной почты имеет вид *adress@tut.by*. Справа от символа @ находится доменное имя - обычный Интернет - адрес компьютера или *Web – сайта* (*www.tut.by*), в котором зарегистрирован пользователь. Слева от символа @ помещается имя почтового ящика пользователя. Для получения собственного почтового ящика необходимо: выбрать сайт, предоставляющий почтовые услуги (например, *http://www.mail.ru/*),; зарегистрироваться в качестве нового пользователя, сообщив некоторые сведения о себе. Результатом регистрации является создание личного почтового ящика, доступ к которому обеспечивается через выбранные Вами ***Имя пользователя (Login) и Пароль*** .

В Интернет для работы с электронной почтой используются прикладные протоколы SMTP и POP3. Протокол SMTP (Simple Mail Transfer Protocol - простой протокол передачи почты) поддерживает передачу сообщений между почтовыми серверами Интернет. Протокол SMTP допускает использование различных сетевых транспортных служб и почтовых серверов и позволяет группировать сообщения в адрес одного получателя и размножать копии e-mail сообщения для передачи в разные адреса.

Почтовый сервер отвечает за прием всех сообщений, идущих на определенное доменное имя. Для каждого пользователя (почтового ящика) на сервере заводится специальный файл, в который помещаются поступающие сообщения в ожидании того, когда пользователь соединится с сервером для их получения. Также сервер производит рассылку адресатам сообщений, поступивших от его пользователей. Предварительно он выясняет, какой сервер отвечает за прием сообщений на доменное имя адресата (например, сообщения с адресами кто-либо@tut.by принимаются сервером tut.by), после чего устанавливает с этим сервером связь по протоколу SMTP.

POP3 (Post Office Protocol) дает пользователю доступ к пришедшим к нему электронным сообщениям, т.е. осуществляет связь между компьютером пользователя и почтовым сервером, на котором зарегистрирован почтовый ящик пользователя. При конфигурировании почтовой программы нужно указать интернет-адрес (доменное имя) почтового сервера для входящих (POP-server) и исходящих (SMTP-server) сообщений (как правило, это один и тот же сервер) и имя пользователя на сервере (login name, POP-user login, учетная запись). При установлении соединения будет запрошен пароль. Имя пользователя и пароль предоставляются администрацией сервера. Пользователь может изменить пароль.

Получение электронной почты можно осуществлять с любого компьютера. Нужно только, чтобы компьютер был подключен к Интернету. Для работы с электронной почтой созданы специальные программы, например, Netscape Mail, Eudora, MS Outlook.

7.3.2. ПОИСК ИНФОРМАЦИИ В СЕТИ ИНТЕРНЕТ

Одной из проблем работы в Интернет является поиск данных. Вследствие большого объема информации, найти необходимые данные можно только с использованием специальных поисковых систем. Чтобы получить доступ к системам поиска, необходимо нажать кнопку Поиск на панели инструментов приложения Internet Explorer (см. 7.2.2). В левой части обозревателя появится панель поиска. Затем в поле Поиск нужно ввести ключевое слово или фразу и нажать кнопку Поиск.

Поиск нужной информации можно проводить с помощью Web-страницы поисковой системы. Сегодня в Internet насчитывается более 200 поисковых систем, которые признаны популярными; общее же их число превышает 1000. Самыми распространенными поисковыми системами являются следующие: русскоязычные: ***www.all.by***, ***www.yandex.ru***, ***www.rambler.ru*** (рис.7.2) , ***www.aport.ru***, ***www.google.ru***, ***www.google.by;*** международные: ***www.yahoo.com, www.excite.com, www.altavista.com, www.lycos.com.***

В поисковых системах поиск ведется в гигантской базе данных URL. Для осуществления успешного поиска важно правильно сформулировать запрос путем задания фразы, одного или нескольихо ключевых слов, которые лучше всего описывают предмет поиска. Некоторые поисковые системы чувствительны к регистру поиска. Поэтому при вводе ключевых слов и фраз необходимо учитывать эту особенность. В поисковых системах можно использовать следующие способы поиска.

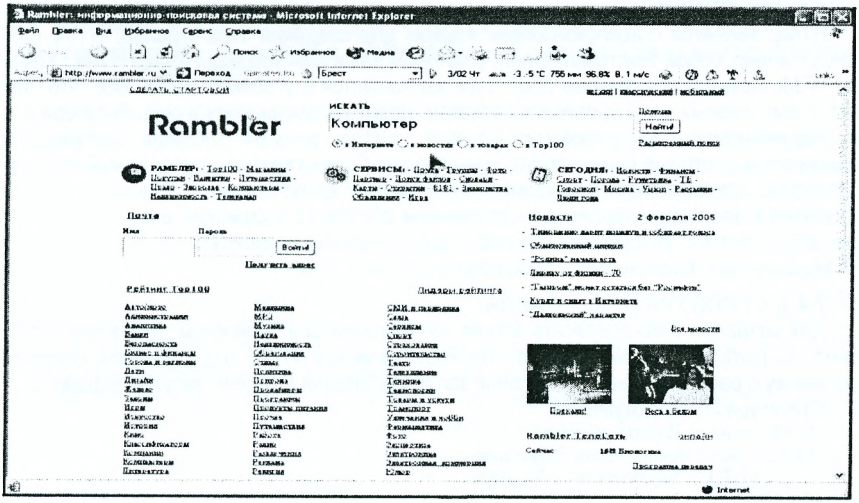


Рис. 7.2. Интерфейс поисковой системы Rambler

Поиск по одному слову. В поле запроса вводится одно или несколько слов, которые могут характеризовать содержание документа. Например, ввести слово **компьютер,** после чего запускается процесс поиска.

Поиск по группе слов. Результат зависит от того, как эти слова введены в конкретной поисковой системе. Например, по запросу **монтаж компьютер** в системе поиска будут разыскиваться документы, в которых встречаются введенные слова. Среди результатов такого поиска возможны нестрогие соответствия, например: ...**монтаж компьютера**...; ...**после монтажа в компьютере** ... и т.д.

Поиск в найденном. В системах Rambler, Aport и Yandex после выполнения поиска можно задать другое ключевое слово (или слова) и включить флажок ***"Искать в найденном"***. Поиск по новым ключевым словам производится только среди ранее найденных документов.

Логическая операция AND. Связывает два элемента запроса. Будут найдены те страницы, на которых есть оба элемента.

Логическая операция OR. Связывает два элемента запроса. Ведется поиск страниц, на которых есть хотя бы один из этих элементов.

После перехода на Web-страницу, можно найти на ней определенный текст, выбрав команду ***Правка - Найти***.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какую структуру имеет адрес электронной почты?
2. Как осуществляется доступ к личному почтовому адресу?
3. Какие прикладные протоколы используются для работы с электронной почтой?
4. Перечислите наиболее распространенные русскоязычные поисковые системы?
5. Какие существуют способы поиска информации в поисковых системах?
6. Какие логические операции используются при поиске информации?

7.4. ЯЗЫК HTML

HTML является общепризнанным языком для создания Web-страниц. HTML-файл представляет собой текстовый файл, в котором записаны команды языка HTML. Команды HTML называются *тэгами (tag)*. Тэги заключаются в угловые скобки. Все, что находится вне угловых скобок, является текстом, подлежащим выводу в окно броузера с теми параметрами форматирования (размер шрифта, элемент таблицы, отступы, центровка и т.п.), которые были установлены тэгами. Существует международный стандарт, полностью описывающий все возможные тэги и их допустимые сочетания. К счастью, программы просмотра терпимы к нарушениям стандарта и ошибкам в синтаксисе: если они встречают тэг, который им незнаком, они его просто игнорируют. Файл, содержащий HTML-документ, должен иметь расширение .htm или .html

7.4.1. СТРУКТУРА HTML - ФАЙЛА

Для создания Web-документа можно воспользоваться блокнотом. Для этого необходимо: открыть текстовый редактор - блокнот: записать HTML- код странички; сохранить страничку с расширением htm или html; загрузить Internet Explorer; запустить файл.

Структура HTML-документа:

<HTML> начало Web-документа
<HEAD> заголовочная часть документа
<TITLE> заголовок окна броузера
</HEAD>
< BODY bgcolor="white">
тело документа, bgcolor=white определяет цвет фона (белый)
.....
</BODY>
</HTML>

Например текст

```
<html>
<head>
<title> первый пример </title>
</head>
<body>
Здравствуйте, это пример первой Web-страницы.
<br>
<b>
При изучении HTML
</b>
</body>
</html> (пример.)
```

создаст WEB-страницу с текстом

Здравствуйте, это пример первой Web-страницы.
При изучении HTML (пример.)

Многие тэги парные (наподобие открывающих и закрывающих скобок). Закрывающему тэгу предшествует символ "/". Действие тэга распространяется на то, что находится между открывающим и закрывающим тэгами. Например, если вы хотите выделить слова полужирным шрифтом, вы пишете <B>слово и еще слова </B>. Здесь <B> и </B> - флажки, указывающие программе просмотра, с какого и до какого места выводить текст жирным шрифтом. Большие и маленькие буквы не различаются, например
,
 и
 совершенно равноправны и одинаково вызывают принудительный перевод строки в тексте.

Программа просмотра (броузер) заменяет все последовательно идущие символы пробелов, табуляции и перевода строки на единственный пробел. Т.е., если создатель web-страницы попытается сделать отступ в несколько пробелов или перейти на новую

строку, не используя специальных тэгов, отступ будет сокращен до одного пробела и все будет воспринято как одна строка. Если что-то изменено в *.htm (html) документе, то, чтобы посмотреть как это выглядит в Internet Explorer, надо не забыть нажать в Internet Explorer кнопку ***Обновить***.

7.4.2. ТЕГИ, НАИБОЛЕЕ ЧАСТО ИСПОЛЬЗУЕМЫЕ В HTML

Кроме уже рассмотренного тэга <B></B> Bold (полужирный), можно рассмотреть другие тэги. Часто используются теги, управляющие параметрами шрифта и форматированием текста: <I></I> Italic (курсив); <P> Paragraph (абзац) - вызывает принудительный перевод строки и отступ в одну пустую строку;
 принудительный перевод строки; <H1> </H1>, <H2> </H2>, ... <H6> </H6> заголовки разных размеров. Заголовок представляет собой полужирный текст с отступами снизу и сверху. Заголовок по умолчанию не центруется; <CENTER> </CENTER> - размещение по центру; <PRE> </PRE> - Preformatted, текст внутри этого тэга выводится в строгом соответствии с тем, как он напечатан в HTML-файле, т.е. выводятся все пробелы, табуляции, отступы и переводы строк. При этом текст выводится шрифтом с фиксированной шириной символов (типа Courier).

<FONT COLOR='green" SIZE=5> </FONT> - установка цвета и размера букв текста. Для установки цвета можно воспользоваться данными из табл. 7.1.

Таблица 7.1 Цветовая палитра, используемая в HTML

Название	*Значение*	*Название*	*Значение*
черный	*Black*	каштановый	*Maroon*
зеленый	*Green*	красный	*Red*
серебряный	*Silver*	синий	*Blue*
лимонный	*Lime*	пурпурный	*Purple*
серый	*Gray*	бирюзовый	*Teal*
оливковый	*Olive*	фуксиновый	*Fuchsia*
белый	*White*	голубой	*Aqua*
желтый	*Yellow*		

Для того чтобы вставить изображение из файла, необходимо использовать тэг <IMG SRC="filename.gif"> - вставить изображение из файла filename.gif. Поддерживаются форматы gif и jpg. Для создания ссылки на другой сайт применяется тэг <A HREF="http://www.website.by"> ссылка </A> - сделать слово "ссылка" гипертекстовой ссылкой на сервер www.website.by.

Рассмотрим пример HTML – кода.

```
<html>
<head>
<title>Мой первый шаг </title>
</head>
<body>
<center>
<br>
<FONT COLOR="red" SIZE=5> <H2>Второй пример HTML.</H2></FONT>
<font color="green"> Добро пожаловать!</font> :) </center>
</br>
<p align="justify">
<img src="my.gif">Пример домашней странички <b>для виртуального друга :)</b>
</p>
</body>
</html>
```

Вид Web- страницы приведен на рис. 7.3

7.4.3. ТАБЛИЦЫ

Таблица состоит из рядов, каждый из которых включает набор элементов.

Элемент таблицы может содержать текст, рисунок, другую таблицу и т.п., то есть все то, что может содержать сам документ. В простейшем случае таблица - это традиционно понимаемая таблица из текстовых ячеек. Рассмотрим тэги, используемые для создания таблицы.

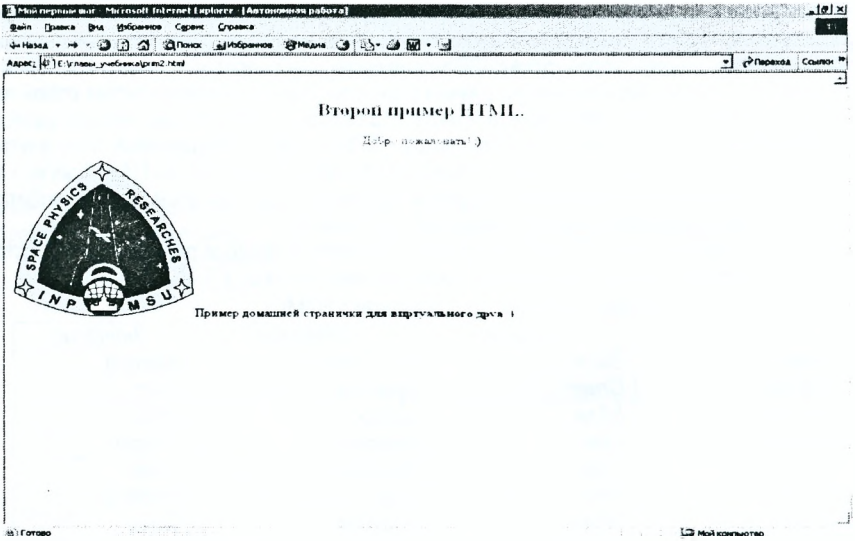


Рис. 7.3. Пример Web - страницы

<TABLE WIDTH=100% BORDER=0> </TABLE> — начало и конец таблицы. Параметр WIDTH определяет ширину таблицы в процентах от ширины окна броузера либо в пикселах, если не указан знак %. Параметр BORDER определяет толщину линий обрамления ячеек таблицы в пикселах. Нулевое значение - нет обрамления. Любой из параметров может отсутствовать.

<TR> </TR> - начало и конец ряда таблицы, должен находиться внутри тэга <TABLE></TABLE>.

Мониторы	***Цена***	***Количество***
Samsung	***150$***	***10***
LG	***145$***	***20***

<TD WIDTH=20% ALIGN=left VALIGN=middle BGCOLOR="blue"> </TD> - определяет ячейку таблицы, которая должна находиться внутри тэга <TR></TR>. Параметр WIDTH определяет ширину ячейки в процентах от ширины таблицы или пикселах, если не указан знак % (по умолчанию устанавливается равная ширина ячеек). ALIGN и VALIGN определяют выравнивание содержимого внутри ячейки по горизонтали и вертикали соответственно, значения ALIGN: left, center, right, значения VALIGN: top, middle, bottom. BGCOLOR определяет цвет фона ячейки. Любой из параметров может отсутствовать.

Рассмотрим пример.

```
<html>
<head>
<title>Прайс-лист </title>
</head>
<body>
<center>
<table BORDER=2 width=60%>
<tr align=center>
<td width=33% ><b><i>Монитор</b></i></td>
<td width=34% > <b><i>Цена</b></i></td>
<td width=33% > <b><i>Количество</b></i></td>
</TR>
<tr align=center>
<td width=33% > <b><i>Samsung</b></i></td>
<td width=34% ><b><i>150$</b></i></td>
<td width=33% ><b><i>10</b></i></td>
</TR>
<tr align=center>
<td width=33% ><b><i>LG</b></i></td>
<td width=34% ><b><i>145$</b></i></td>
<td width=33% ><b><i>20</b></i></td>
</TR>
</TABLE>
</CENTER>
</body>
</html>
```

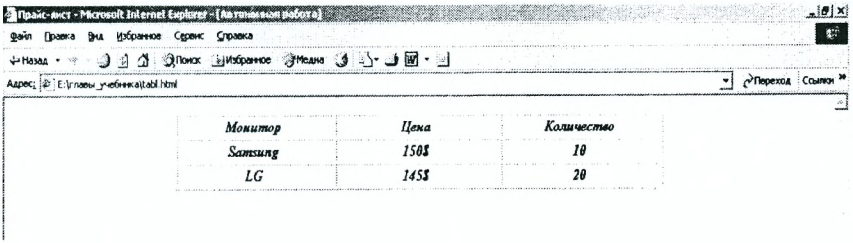


Рис. 7.4. Web – страница с таблицей

Вид Web – страницы данного кода приведен на 7.4.

7.4.4. РЕГИСТРАЦИЯ СТРАНИЦЫ

После создания Web - страницы, ее необходимо разместить на каком-либо сервере, чтобы она была доступна через Интернет. Перед тем как размещать на сервере страницу, необходимо ознакомиться с правилами размещения, которые приводятся прямо на выбранном сервере. Существует ряд бесплатных серверов, на которых можно разместить Web – страницы. Среди них можно выделить русскоязычные: http://www.by.ru/; http://www.tut.by/; http://www. chat.ru/;http://www.halyva.ru/; англоязычные: http://www.geocites.com/; http://www.hypermart.net/; http://www.angelfire.com/.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что представляет собой файл HTML?
2. Какие расширения может иметь файл, содержащий HTML – документ?
3. Как называются команды HTML?
4. Каким символом закрываются парные тэги?
5. Какие тэги позволяют установить полужирный шрифт и курсив?
6. Как установить цвет и размер букв текста?
7. Как вставить изображение из файла в Web – страницу?
8. Какие тэги используются для формирования таблицы?
9. С помощью какого тэга можно регулировать толщину линий обрамления ячеек таблицы?
10. Какие русскоязычные сайты можно использовать для регистрации Web – странички?

ЗАКЛЮЧЕНИЕ

Интернет в настоящее время является наиболее доступным средством обмена информацией. Объем информации, содержащейся в Интернет, практически неограничен, а постоянное пополнение и обновление этой информации делает его универсальным источником данных во всех областях человеческой деятельности. В данном разделе рассмотрены вопросы организации работы в Интернет: поиск информации, работа с электронной почтой. В заключении даны основы HTML, что позволит будущему инженеру создавать WEB – странички для активной рекламы результатов собственных исследований.

ЛИТЕРАТУРА

К разделу 1

1. Алексеев А. П., Информатика, - М.: "Солон-Р", 2002. – 400 с., ил.
2. Берлинер Э. М. Глазырина И. Б., Глазырин Б. Э. Windows 2000 Proffessional: Русская и английская версия. – М.: Компьютер Пресс, 2000. – 360 с.: ил.
3. Буза М. К., Певзнер Л. В. Windows – приложения: от операции к реализации: Учебное пособие. – Мн..Высш. шк., 1998. – 490 с.: ил.
4. Быков В. Л. Основы информатики: конспект лекций. – Брест: БГТУ, 2002. – 253 с.
5. Информатика/ под ред. Н. В. Макаровой. – М.: Финансы и статистика, 2000. – 768 с.: ил.
6. Информатика. Базовый курс / Симонович С. В. и др. – СПб: Питер, 2000. – 640 с.: ил.
7. Использование Microsoft Windows XP Home Edition. Специальное издание.: - М.: Издательский дом "Вильямс", 2002. – 896 с.: ил.
8. Коуров Л. В., Словарь-справочник по информатике, - Мн.: Амалфея, 2000. – 176 с.
9. Леонтьев В. П., Новейшая энциклопедия персонального компьютера 2000. – 2-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. Новейший самоучитель работы на компьютере. - Москва: издательство "ДЕСС КОМ", 2000. - 654 с.
12. Основы информатики: Учебное пособие; под ред. А. Н. Морозевича. – Мн.: Новое знание, 2001. – 544 с.: ил.
13. Острейковский В. А. Информатика. – М.: "Высшая школа", 2000.-511 с.: ил.
14. Олифер Н. А., Олифер В. Г. Сетевые операционные системы. Центр информационных технологий. – Брест: БГТУ, 2005.
15. Фигурнов В. Ф. IBM PC для пользователя. - М.: Финансы и статистика, 2003.
16. Экономическая информатика / под ред. П. В. Конюховского и Д. Н. Колесова. – СПб.: Питер, 2001. – 560 с.: ил.

К разделу 2

17. Математический энциклопедический словарь/ Гл. ред. Ю.В. Прохоров; ред. кол.: С. И. Адян, Н.С. Бахвалов, В.И. Ботюцков, А.П. Ершов, Л.Д. Кудрявцев, А.Л. Онищик, А.П. Юшкевич. – М.: Сов. Энциклопедия, 1988. – 847 с.,ил.
18. Гурский Д.А. Вычисления в MathCad – Мн.: Новое знание, 2003. – 814 с.
19. Mathcad 6.0 PLUS. Финансовые, инженерные и научные расчеты в среде Windows 95 /Перевод с англ. – М.: Информационно – издательский дом «Филинъ», 1996 – 712 с.
20. Численные методы решения задач строительной механики: /Справ. пособие / В.П. Ильин, В.В. Карпов, А.М. Масленников; Под общ. Ред. В.П. Ильина, - Мн.: Выш. шк., 1990. -349 с.
21. Кулаичев А.П. Методы и средства анализа данных в среде Windows. Stadia6.0. Изд. 2-е. перераб. и доп. - М.: Информатика и компьютеры,1998. - 270 с.
22. Кирьянов Д. В. Самоучитель MathCad 11. – СПб.: БХВ-Петербург, 2003.- 560 с.: ил.
23. Рыжиков Ю. И. Решение научно-технических задач на персональном компьютере. – СПб.: КОРОНА принт, 2000. – 272 с.

См. 9.
24. 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.
25. Турчак Л. И. Основы численных методов. – М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 320 с.

К разделу 4

26. Браун С. Visual Basic 6: учебный курс. – СПб.: Питер, 2002. – 576 с. ил.
27. Быков В. Л. Основы программирования на языке VB 6.0. – Брест: БГТУ, 2002. - 230 с.
28. Волчёнков Н. Г. Программирование на Visual Basic 6. – М.: ИНФРА, 2000.
29. Гарри Корнель . Программирование в среде VB5, - Мн.:ООО "Папури", 1998.-608 с.:ил.
30. Михаель Рейтингу, Геральд Муч. Visual Basic 6.0-К.:Издательская группа BHV, 2000. - 288 с.: ил.
31. Microsoft Corporation. разработка приложений на VB 6.0 - М.: Издательско-торговый дом "Русская Редакция", 2000. - 400 с.ил.
32. Брайн Сайлер и Джефф Скоттс. Использование VB 6.0 - М.: СПб.; К.: Издательский дом "Вильямс", 2001.- 832 с.: ил.

К разделу 5

См. раздел 1 п. п. 2, 3, 4, 6, 11, 12.

К разделу 6

См. раздел 1 п. п. 11, 12.

33. Гарнаев А.Ю. Использование MS EXCEL и VBA в экономике и финансах. – СПб.: БХВ – Санкт- Петербург, 1999. – 336 с.
34. Гельман В. Я. Решение математических задач средствами Excel: Практикум. – СПб.: Питер, 2003. – 240 с.: ил.
35. Орвис Вильям Excel для ученых, инженеров и студентов: Пер с англ. – К.: Юниор, 1999. – 528 с.: ил.

К разделу 7

См. раздел 1 п. п. 10, 11, 13, 14

36. Ноэль Истабрук Internet. Освой самостоятельно за 24 часа / Пер. с англ. – М.: ЗАО «Издательство БИНОМ», 1998. - 320 с.: ил.
37. Холмогоров В. Основы Web – мастерства. Учебный курс. – СПб.: Питер, 2002. - 352 с.: ил.

Учебное издание

Быков Вячеслав Леонидович
Ашаев Юрий Павлович

Основы информатики

Допущено Советом Брестского государственного технического университета в качестве пособия для студентов технических специальностей

Ответственный за выпуск ***Быков В.Л.***
Редактор ***Строкач Т.В.***
Компьютерная верстка ***Боровикова Е.А.***
Корректор ***Никитчик Е.В.***

Лицензия № 02330/0133017 от 30.04.2004 г.
Подписано к печати 22.02.2006 г.
Формат 60x84 1/16. Бумага «Океан». Гарнитура Arial Narrow.
Усл. печ. л. 25. Уч.-изд. л. 26,9. Заказ № 139. Тираж 200 экз.
Отпечатано на ризографе учреждения образования
«Брестский государственный технический университет».
Лицензия № 02330/0148711 от 30.04. 2004 г.
224017, г. Брест, ул. Московская, 267