

А. В. МАЙСТРЕНКО

ИНФОРМАТИКА



♦ Издательство ТГТУ ♦

УДК 007(075)
ББК з 81я73-1
М 149

Рецензенты:
Заведующий кафедрой компьютерного и математического
моделирования ТГУ им. Г. Р. Державина,
доктор технических наук, профессор
А. А. Арзамасцев
Заведующий кафедрой САПР ТГТУ,
кандидат технических наук, доцент
И. В. Милованов

А. В. Майстренко

М 149 Информатика: Учеб. пособие. Тамбов: Изд-во Тамб. гос. тех. ун-та, 2003. Ч. II. 80 с.
ISBN 5-8265-0161-8

Во второй части учебного пособия продолжено рассмотрение основных разделов информатики, определяющих базовый уровень подготовки инженеров: системного программного инструментария, основных офисных и сетевых информационных технологий, основ алгоритмизации и программирования.

Пособие предназначено для студентов направлений 655400 – "Энерго- и ресурсосберегающие процессы в химической технологии", 65500 – "Биотехнология" и 655800 – "Пищевая инженерия" всех форм обучения.

УДК 007(075)
ББК з 81я73-1

ISBN 5-8265-0161-8

Министерство образования Российской Федерации
Тамбовский государственный технический университет

А. В. МАЙСТРЕНКО

ИНФОРМАТИКА

ЧАСТЬ II

Утверждено Ученым советом университета
в качестве учебного пособия

Тамбов
Издательство ТГТУ
2003

Учебное издание

МАЙСТРЕНКО Александр Владимирович

ИНФОРМАТИКА

Учебное пособие

Редактор Т. М. Федченко
Компьютерное макетирование М. А. Филатовой

Подписано в печать 23.01.2003
Формат 60 × 84 / 16. Бумага офсетная. Печать офсетная.
Гарнитура Times New Roman. Объем: 4,65 усл. печ. л.; 4,50 уч.-изд. л.
Тираж 100 экз. С. 46

Издательско-полиграфический центр
Тамбовского государственного технического университета,
392000, Тамбов, Советская, 106, к. 14

ВВЕДЕНИЕ

Войти в XXI век образованным человеком можно, только хорошо владея информационными технологиями. Ведь деятельность людей все в большей степени зависит от их информированности, способности эффективно использовать информацию. Для свободной ориентации в информационных потоках современный специалист любого профиля – инженер, экономист, юрист, социолог, журналист и др. – должен уметь получать, обрабатывать и использовать информацию с помощью компьютеров, телекоммуникаций и других средств связи. Специалисту, чтобы на должном уровне выполнять свои обязанности, необходимы инструментарий и методология его применения для обработки информации. Это сравнимо с использованием средств передвижения: теоретически человек может пешком преодолеть любое расстояние, но современный темп жизни просто немыслим без применения автомобиля, поезда, самолета и т.д. То же самое происходит и в области, связанной с обработкой информации: теоретически человек сам может переработать без компьютера любую информацию, но сделает это эффективнее, если овладеет соответствующими знаниями и умениями. Заложить фундамент информационной культуры призвана дисциплина "Информатика".

Данное учебное пособие продолжает знакомить читателя с основами "Информатики". В первой части были рассмотрены вопросы, связанные с ключевыми понятиями информации, информационных систем и технологий, рассмотрено техническое и программное обеспечение персональных компьютеров, изучены основы функционирования операционных систем и файловая структура организации данных в ПК.

В настоящем пособии рассматриваются основные офисные компьютерные технологии и системное программное обеспечение, основы организации и функционирования компьютерных сетей, основы алгоритмизации.

6 ОСНОВНЫЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

Работа с текстом

При описании текстовых процессоров (прикладное программное обеспечение, используемое для создания текстовых документов) следует выделить две группы программных продуктов этого типа. Первая группа ориентирована на создание документов разной степени сложности с мощными средствами форматирования и включения графики. Типичным представителем этой группы является текстовый процессор Microsoft Word. Вторая группа текстовых процессоров (их часто называют текстовыми редакторами) ориентирована для работы с чисто текстовыми файлами, среди которых могут быть тексты программ, написанные на различных языках, конфигурационные файлы, файлы настройки и другие. Ярким представителем таких программных продуктов является MultiEdit версий, начиная с 5.0. Этот текстовый редактор имеет мощную систему контекстной замены, встроенный язык макрокоманд на уровне Visual Basic, средства поддержки внутренней среды, средства помощи при наборе ключевых слов.

Редакторы текстов программ

Редакторы текстов программ рассчитаны на редактирование программ на том или ином языке программирования. Часто они встроены в систему программирования на некотором языке программирования, и непосредственно из них можно запускать программы на компиляцию и выполнение. Примером могут служить редакторы, встроенные в системы программирования Turbo C, Turbo Pascal и т.д., а также MultiEdit и Brief.

Редакторы, рассчитанные на тексты программ, как правило, выполняют следующие функции:

- диалоговый просмотр текста;
- редактирование строк программы;
- копирование и перенос блоков текста;
- копирование одной программы или ее части в указанное место другой программы;
- контекстный поиск и замену подстрок текста;
- автоматический поиск строки, содержащей ошибку;
- распечатку программы или ее части.

Кроме того, часто редакторы текстов программ позволяют автоматически проверять синтаксическую правильность программ. Иногда эти редакторы объединены с отладчиками программ на уровне исходного текста.

Редакторы текстов программ, как правило, можно использовать для создания и корректировки небольших документов. Однако для серьезной работы с документами лучше использовать редакторы, ориентированные на работу с документами.

Редакторы документов

Программы для обработки документов в отличие от редакторов текстов программ ориентированы на работу с текстами, имеющими структуру документа, т.е. состоящими из разделов, страниц, абзацев, предложений, слов и т.д. Поэтому редакторы для обработки документов могут обеспечивать функции, ориентированные на структуру документа, а именно:

- возможность использования различных шрифтов символов;
- работу с пропорциональными шрифтами;
- задание произвольных межстрочных промежутков;
- автоматический перенос слов на новую строку;
- автоматическую нумерацию страниц;
- обработку и нумерацию сносок;
- печать верхних и нижних заголовков страниц (колонтитулов);
- выравнивание краев абзаца;
- набор текста в несколько столбцов;
- создание таблиц и построение диаграмм;
- проверку правописания и подбор синонимов;
- построение оглавлений, индексов и т.д.

Всего существует несколько сотен редакторов текстов, от самых простых до весьма мощных и сложных. Среди наиболее распространенных в мире редакторов назовем Microsoft Word, Write, WordPerfect.

Тем пользователям, которым требуется обеспечить высокое качество напечатанных документов или подготавливать документы большого объема, рекламные буклеты или книги необходимо воспользоваться мощным редактором документов – текстовым процессором *Microsoft Word*. Вот некоторые из функций, поддерживаемых Word:

- использование множества различных шрифтов (размеров и начертаний) символов и различных способов их выделения (полужирные, курсивные, подчеркнутые символы и т.д.);
- указание параметров абзацев текста и страниц документа;
- набор текста в несколько колонок;
- печать верхних и нижних колонтитулов произвольного вида;
- автоматическое формирование оглавления и различных видов указателей;
- различное оформление таблиц и абзацев;
- включение рисунков (графических файлов);
- макетирование документа в любом виде;
- вставка формул и диаграмм;
- проверка орфографии и пунктуации.

Опытные пользователи очень ценят такую возможность Word как стили. Word позволяет записать в так называемую таблицу стилей все параметры наиболее часто используемых видов оформления текста: абзацев, символов и разделов документа. Если сделать это, то затем любому участку текста можно будет присвоить один из "стандартных" видов оформления с помощью одного-двух нажатий клавиш. Это не только значительно ускоряет набор документа, но и повышает гибкость его оформления. Например, чтобы изменить шрифт и расположение у всех заголовков некоторого уровня (скажем, заголовков параграфов), не потребуется искать эти заголовки и вручную изменять их форматирование – достаточно исправить стиль для этих заголовков, и они автоматически примут нужное оформление.

В одно из многих полезных свойств Word входит автоматическая корректура текста по границам, автоматический перенос слов и правка правописания слов, сохранение текста в определенный устанавливаемый промежуток времени, наличие мастеров текстов и шаблонов, позволяющих в считанные ми-

нуты создать деловое письмо, факс, автобиографию, расписание, календарь и многое другое. Word обеспечивает поиск заданного слова или фрагмента текста, замену его на указанный фрагмент, удаление, копирование во внутренний буфер или замену по шрифту, гарнитуре или размеру шрифта, а так же по надстрочным или по подстрочным символам. Наличие закладки в тексте позволяет быстро перейти к заложенному месту в тексте. Можно так же автоматически включать в текст дату, время создания, обратный адрес и имя написавшего текст. При помощи макрокоманд Word позволяет включать в текст базы данных или объекты графики, музыкальные модули в формате *.wav. Для ограничения доступа к документу можно установить пароль на текст, который Word будет спрашивать при загрузке текста для выполнения с ним каких-либо действий. Word позволяет открывать много окон для одновременной работы с несколькими текстами, а так же разбить одно активное окно по горизонтали на несколько и выровнять их.

Word позволяет работать одновременно с несколькими документами одновременно, что упрощает работу с документами.
В данную программу встроен редактор векторной графики. Векторные графические объекты – это различные схемы, графики, рисунки, выноски и другие. Объекты могут размещаться в любом месте документа, могут изменять свое местоположение, размер, формат, независимо от текста. Word располагает хорошими возможностями для создания и оформления произвольных таблиц.

Внешний вид программы Microsoft Word (рис. 1) является характерным для всех программ, написанных под операционную систему Windows.

Среди простых редакторов текста в России наибольшее распространение получил *Лексикон*. Он имеет интерфейс на русском языке и позволяет подготавливать несложные документы с текстом на русском и английском языках. Лексикон удачно заполняет свою "экологическую нишу" – он вполне подойдет тем, кому нужен простой инструмент для подготовки небольших и несложных документов, не требующих высокого полиграфического качества.

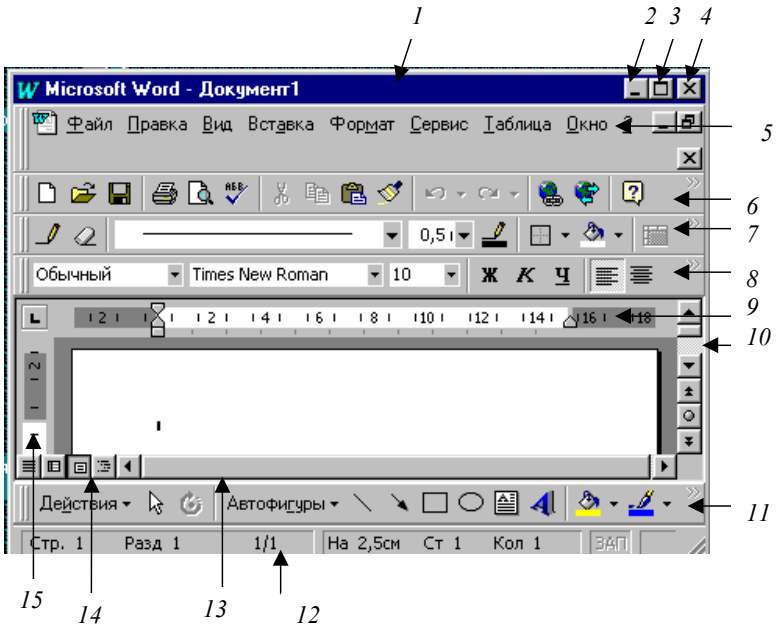


Рис. 1 Внешний вид окна текстового процессора Microsoft Word:
1 – наименование документа; *2* – свернуть окно; *3* – развернуть окно;
4 – закрыть окно; *5* – главное меню; *6* – панель инструментов стандартная;
7 – панель инструментов таблицы и границы; *8* – панель инструментов форматирования; *9* – горизонтальная линейка форматирования;
10 – вертикальная полоса прокрутки; *11* – панель инструментов рисования;
12 – строка состояния; *13* – горизонтальная полоса прокрутки; *14* – различные виды просмотра; *15* – вертикальная линейка форматирования

Работа с электронными таблицами

Электронная таблица – самая распространенная и мощная информационная технология для профессиональной работы с данными. Для управления электронной таблицей созданы специальные программные продукты – табличные процессоры. Имеющиеся сегодня на рынке табличные процессоры способны

работать в широком круге экономических приложений и могут удовлетворить практически любого пользователя.

Идея создания электронной таблицы возникла у студента Гарвардского университета (США) Дэна Бриклина (Dan Bricklin) в 1979 году. Выполняя скучные вычисления экономического характера с помощью бухгалтерской книги, он и его друг Боб Франкстон (Bob Frankston), который разбирался в программировании, разработали первую программу электронной таблицы, названную ими *VisiCalc*.

VisiCalc скоро стала одной из наиболее успешных программ. Первоначально она предназначалась для компьютеров типа Apple II, но потом была трансформирована для всех типов компьютеров. Многие считают, что резкое повышение продаж компьютеров типа Apple в то время и было связано с возможностью использования на них табличного процессора VisiCalc. В появившихся позже электронных таблицах-аналогах (например, *SuperCalc*) основные идеи VisiCalc были многократно усовершенствованы.

Новый существенный шаг в развитии электронных таблиц – появление в 1982 году на рынке программных средств *Lotus 1-2-3*. Lotus был первым табличным процессором, интегрировавшим в своем составе, помимо обычных инструментов, графику и возможность работы с системами управления базами данных. Поскольку Lotus был разработан для компьютеров типа IBM, он сделал для этой фирмы то же, что VisiCalc в свое время сделал для фирмы Apple. После разработки Lotus 1-2-3 компания Lotus в первый же год повышает свой объем продаж до 50 млн. дол. и становится самой большой независимой компанией – производителем программных средств. Успех компании Lotus привел к ужесточению конкуренции, вызванной появлением на рынке новых электронных таблиц, таких, как VP Planner компании Paperback Software и Quattro Pro компании Borland International, которые предложили пользователю практически тот же набор инструментария, но по значительно более низким ценам.

Следующий шаг – появление в 1987 году табличного процессора Excel фирмы Microsoft. Эта программа предлагала более простой графический интерфейс в комбинации с ниспадающими меню, значительно расширив при этом функциональные возможности пакета и повысив качество выходной информации. Расширение спектра функциональных возможностей электронной таблицы, как правило, ведет к усложнению работы с программой.

Разработчикам Excel удалось найти золотую середину, максимально облегчив пользователю освоение программы и работу с ней. Благодаря этому Excel быстро завоевала популярность среди широкого круга пользователей. В настоящее время, несмотря на выпуск компанией Lotus новой версии электронной таблицы, в которой использована трехмерная таблица с улучшенными возможностями, Excel занимает ведущее место на рынке табличных процессоров.

Рассмотрим основы работы табличных процессоров на примере Microsoft Excel.

Итак, электронная таблица – компьютерный эквивалент обычной таблицы, в клетках (ячейках) которой записаны данные различных типов: тексты, даты, формулы, числа. Результат вычисления формулы в клетке является изображением этой клетки. Числовые данные и даты могут рассматриваться как частный случай формул.

Главное достоинство электронной таблицы – это возможность мгновенного пересчета всех данных, связанных формульными зависимостями при изменении значения любого операнда.

Рабочая область электронной таблицы состоит из строк и столбцов, имеющих свои имена. Имена строк – это их номера. Нумерация строк начинается с единицы и заканчивается максимальным числом, установленным для данной программы. Имена столбцов – это буквы латинского алфавита сначала от A до Z, затем от AA до AZ, BA до BZ и т.д.

Максимальное количество строк и столбцов определяется особенностями используемой программы и объемом памяти компьютера. Современные программы дают возможность создавать электронные таблицы, содержащие более одного миллиона ячеек, хотя для практических целей в большинстве случаев этого не требуется.

Пересечение строки и столбца образует *ячейку таблицы,* имеющую свой уникальный адрес. Для указания адресов ячеек в формулах используются *ссылки* (например, A2 и т.п.).

В электронной таблице существует понятие *блока* (диапазона) *ячеек,* также имеющего свой уникальный адрес. В качестве блока ячеек может рассматриваться строка или часть строки, столбец или часть столбца, а также прямоугольник, состоящий из нескольких строк и столбцов или их частей. Адрес блока ячеек задается указанием ссылок первой и последней его ячеек, между которыми, например, ставится разделительный символ – двоеточие ":" или две точки подряд "..".

Каждая команда электронной таблицы требует указания блока (диапазона) ячеек, в отношении которых она должна быть выполнена.

Блок используемых ячеек может быть указан двумя путями: либо *непосредственным набором* с клавиатуры начального и конечного адресов ячеек, формирующих диапазон, либо *выделением* соответ-

ствующей части таблицы при помощи клавиш управления курсором. Удобнее задавать диапазон выделением ячеек.

Типичными установками, принимаемыми по умолчанию на уровне всех ячеек таблицы, являются: ширина ячейки в 9 разрядов, левое выравнивание для символьных данных и основной формат для цифровых данных с выравниванием вправо.

Документ, обрабатываемый табличным процессором Microsoft Excel называется *рабочей книгой*, содержащей несколько *листов,* в которые могут входить таблицы, диаграммы или макросы. Можно создать книгу для совместного хранения в памяти интересующих листов и указать, какое количество листов она должна содержать. Все листы рабочей книги сохраняются в одном файле.

В каждую ячейку электронной таблицы пользователь может ввести данные одного из следующих возможных типов: символьные, числовые, формулы и функции, а также даты.

1 *Символьные (текстовые) данные* имеют описательный характер. Они могут включать в себя алфавитные, числовые и специальные символы.

2 *Числовые данные* не могут содержать алфавитных и специальных символов, поскольку с ними производятся математические операции. Единственными исключениями являются десятичная точка (запятая) и знак числа, стоящий перед ним.

3 *Формулы.* Видимое на экране содержимое ячейки, возможно, – результат вычислений, произведенных по имеющейся, но не видимой в ней формуле. Формула может включать ряд арифметических, логических и прочих действий, производимых с данными из других ячеек. Начинается формула со знака "=".

4 *Функции.* Функция представляет собой программу с уникальным именем, для которой пользователь должен задать конкретные значения аргументов функции, стоящих в скобках после ее имени. Функцию (так же, как и число) можно считать частным случаем формулы. Различают статистические, логические, финансовые и другие функции.

5 *Даты.* Особым типом входных данных являются даты. Этот тип данных обеспечивает выполнение таких функций, как добавление к дате числа (пересчет даты вперед и назад) или вычисление разности двух дат (длительности периода). Даты имеют внутренний (например, дата может выражаться количеством дней от начала 1900 года или порядковым номером дня по Юлианскому календарю) и внешний формат. Внешний формат используется для ввода и отображения дат.

РАБОТА С БАЗАМИ ДАННЫХ

Цель любой информационной системы – обработка данных об объектах реального мира. В широком смысле слова база данных – это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области. Под *предметной областью* принято понимать часть реального мира, подлежащего изучению для организации управления и в конечном счете автоматизации, например, предприятие, вуз и т.д.

Централизованный характер управления данными в базе данных предполагает необходимость существования некоторого лица (группы лиц), на которое возлагаются функции администрирования данными, хранимыми в базе.

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро извлекать выборку с произвольным сочетанием признаков. Сделать это возможно, только если данные структурированы.

Структурирование – это введение соглашений о способах представления данных.

Неструктурированными называют данные, записанные, например, в текстовом файле.

Пользователями базы данных могут быть различные прикладные программы, программные комплексы, а также специалисты предметной области, выступающие в роли потребителей или источников данных, называемые *конечными пользователями.*

В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария – *системы управления базами данных.*

База данных (БД) – это поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Система управления базами данных (СУБД) – это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

КЛАССИФИКАЦИЯ БАЗ ДАННЫХ

По *технологии обработки* данных базы данных подразделяются на централизованные и распределенные.

Централизованная база данных хранится в памяти одной вычислительной системы. Если эта вычислительная система является компонентом сети ЭВМ, возможен распределенный доступ к такой базе. Такой способ использования баз данных часто применяют в локальных сетях ПК.

Распределенная база данных состоит из нескольких, возможно пересекающихся или даже дублирующих друг друга частей, хранимых в различных ЭВМ вычислительной сети. Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

По *способу доступа* к данным базы данных разделяются на базы данных с *локальным доступом* и базы данных с *удаленным (сетевым) доступом.*

Системы централизованных баз данных с сетевым доступом предполагают различные *архитектуры* подобных систем.

1 *Файл-сервер*. Архитектура систем БД с сетевым доступом предполагает выделение одной из машин сети в качестве центральной (сервер файлов). На такой машине хранится совместно используемая централизованная БД. Все другие машины сети выполняют функции рабочих станций, с помощью которых поддерживается доступ пользовательской системы к централизованной базе данных. Файлы базы данных в соответствии с пользовательскими запросами передаются на рабочие станции, где в основном и производится обработка. При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает. Пользователи могут создавать также на рабочих станциях локальные БД которые используются ими монопольно. Концепция файл-сервера условно отображена на рис. 2.

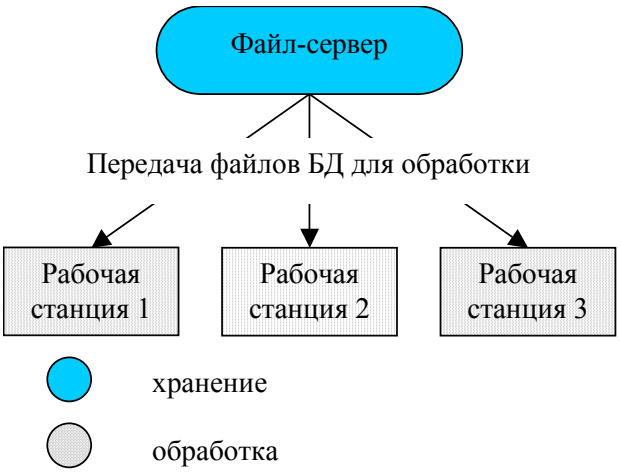


Рис. 2 Схема обработки информации в БД по принципу файл-сервер

2 *Клиент-сервер*. В этой концепции подразумевается, что помимо хранения централизованной базы данных центральная машина (сервер базы данных) должна обеспечивать выполнение основного объема обработки данных. Запрос на данные, выдаваемый клиентом (рабочей станцией), порождает поиск и извлечение данных на сервере. Извлеченные данные (но не файлы) транспортируются по сети от сервера к клиенту. Спецификой архитектуры клиент-сервер является использование языка запросов SQL. Концепция клиент-сервер условно изображена на рис. 3.

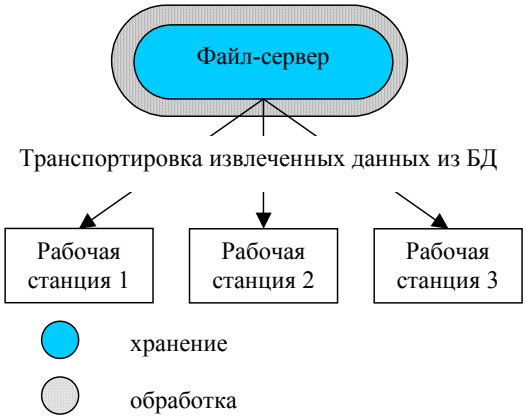


Рис. 3 Схема обработки информации в БД по принципу клиент-сервер

СТРУКТУРНЫЕ ЭЛЕМЕНТЫ БАЗЫ ДАННЫХ

Понятие базы данных тесно связано с такими понятиями структурных элементов, как поле, запись, файл (таблица).

Поле – элементарная единица логической организации данных, которая соответствует неделимой единице информации – реквизиту. Для описания поля используются следующие *характеристики:* имя, тип, длина (определяется максимально возможным количеством символов); точность для числовых данных.

Запись – совокупность логически связанных полей.

Файл (таблица) – совокупность экземпляров записей одной структуры.

В структуре записи файла указываются поля, значения которых являются *ключами: первичными* (ПК), которые идентифицируют экземпляр записи, и *вторичными* (ВК), которые выполняют роль поисковых или группировочных признаков (по значению вторичного ключа можно найти несколько записей).

ВИДЫ МОДЕЛЕЙ ДАННЫХ

Ядром любой базы данных является модель данных. Модель данных представляет собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

Модель данных – совокупность структур данных и операций их обработки.

СУБД основывается на использовании иерархической, сетевой или реляционной модели, на комбинации этих моделей или на некотором их подмножестве.

Иерархическая модель данных

Иерархическая структура представляет совокупность элементов, связанных между собой по определенным правилам. Объекты, связанные иерархическими отношениями, образуют ориентированный *граф* (перевернутое дерево), вид которого представлен на рис. 4.

К основным понятиям иерархической структуры относятся: уровень, элемент (узел), связь. *Узел* – это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. Каждый узел на более низком уровне связан только с одним узлом, находящимся на более высоком уровне. Иерархическое дерево имеет только одну вершину (корень дерева), не подчиненную никакой другой вершине и находящуюся на самом верхнем (первом) уровне. Зависимые (подчиненные) узлы находятся на втором, третьем и т.д. уровнях. Количество деревьев в базе данных определяется числом корневых записей.

К каждой записи базы данных существует только один (иерархический) путь от корневой записи.

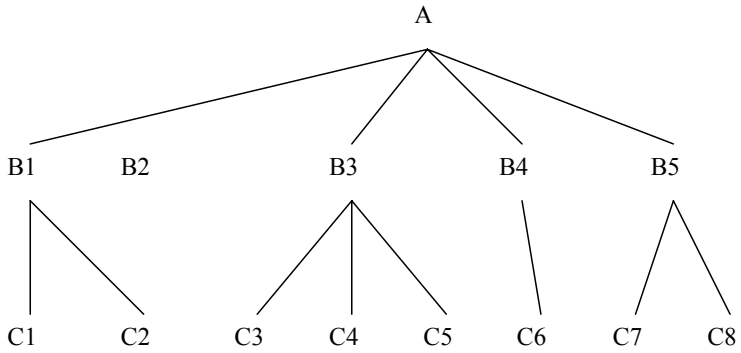


Рис. 4 Графическое изображение иерархической структуры БД

Сетевая модель данных

В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

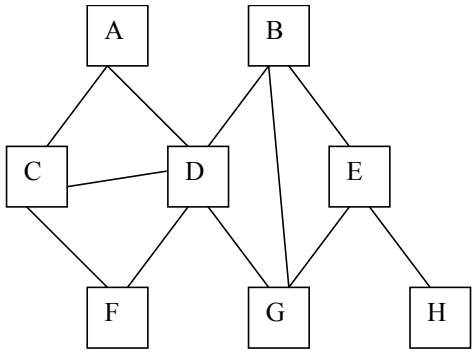


Рис. 5 Графическое изображение сетевой структуры БД

Реляционная модель данных

Понятие реляционный (англ. *relation* – отношение) связано с разработками известного американского специалиста в области систем баз данных Е. Кодда.

Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая *реляционная таблица* представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы – один элемент данных;
- все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- каждый столбец имеет уникальное имя;
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Отношения представлены в виде *таблиц,* строки которых соответствуют *записям,* а столбцы – *полям.*

Поле, каждое значение которого однозначно определяет соответствующую запись, называется *простым ключом* (ключевым полем). Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет *составной ключ.*

Чтобы связать две реляционные таблицы, необходимо ключ первой таблицы ввести в состав ключа второй таблицы (возможно совпадение ключей); в противном случае нужно ввести в структуру первой таблицы *внешний ключ* – ключ второй таблицы.

Обзор СУБД

Системой управления базами данных называют *программную систему,* предназначенную для создания на ЭВМ общей базы данных, используемой для решения множества задач. Подобные системы служат для поддержания базы данных в актуальном состоянии и обеспечивают эффективный доступ пользователей к содержащимся в ней данным в рамках предоставленных пользователям полномочий.

СУБД предназначена для централизованного управления базой данных в интересах всех работающих в этой системе.

По степени универсальности различают два класса СУБД:

- системы общего назначения;
- специализированные системы.

СУБД общего назначения не ориентированы на какую-либо предметную область или на информационные потребности какой-либо группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной системе и поставляется многим пользователям как коммерческое изделие. Такие СУБД обладают средствами настройки на работу с конкретной базой данных. Использование СУБД общего назначения в качестве инструментального средства для создания автоматизированных информационных систем, основанных на технологии баз данных, позволяет существенно сокращать сроки разработки, экономить трудовые ресурсы. Этим СУБД присущи развитые функциональные возможности и определенная функциональная избыточность.

Специализированные СУБД создаются в редких случаях при невозможности или нецелесообразности использования СУБД общего назначения.

СУБД общего назначения – это сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией базы данных информационной системы.

Рынок программного обеспечения ПК располагает большим числом разнообразных по своим функциональным возможностям коммерческих систем управления базами данных общего назначения, а также средствами их окружения практически для всех массовых моделей машин и для различных операционных систем.

Используемые в настоящее время СУБД обладают средствами обеспечения целостности данных и надежной безопасности, что дает возможность разработчикам гарантировать большую безопасность данных при меньших затратах сил на низкоуровневое программирование. Продукты, функционирующие в среде Windows, выгодно отличаются удобством пользовательского интерфейса и встроенными средствами повышения производительности.

Производительность СУБД оценивается:

- временем выполнения запросов;
- скоростью поиска информации в неиндексированных полях;
- временем выполнения операций импортирования базы данных из других форматов;
- скоростью создания индексов и выполнения таких массовых операций, как обновление, вставка, удаление данных;
- максимальным числом параллельных обращений к данным в многопользовательском режиме**;**
- временем формирования отчета.

РАБОТА С ГРАФИЧЕСКИМИ ОБЪЕКТАМИ

Потребность ввода графиков, диаграмм, схем, рисунков в различные документы вызвала необходимость создания графических процессоров. Графические процессоры представляют собой инструментальные средства, позволяющие создавать и модифицировать графические образы с использованием следующих информационных технологий:

- коммерческой графики;
- иллюстративной графики;
- научной графики;
- когнитивной графики и др.

Информационные технологии *коммерческой* или *деловой* графики обеспечивают отображение информации, хранящейся в табличных процессорах, базах данных и отдельных локальных файлах в виде двух- или трехмерных графиков, круговых диаграмм, гистограмм, линейных графиков и других. Позво-

ляют создавать иллюстрации для различных документов в виде регулярных структур (векторная графика) и нерегулярных структур (растровая графика). Так же к этой группе программного обеспечения можно отнести графические процессоры по обработке готовых изображений и создания анимированной графики.

Информационные технологии *научной графики* предназначены для обслуживания задач картографии, оформления научных расчетов, содержащие различные химические, математические формулы и т. п.

Когнитивные графические процессоры представляют собой комплекс виртуальных устройств, программ и систем, реализующих совокупную обработку зрительной информации в виде образов, процессов, структур и позволяющих средствами диалога реализовать методы и приемы представления условий задачи или подсказки в виде зрительных образов. Виртуальное устройство является функциональным эквивалентом устройства, предоставляемого пользователю, независимо от того, имеется ли данное устройство в системе или нет. Когнитивная графика позволяет образно представить различные математические формулы и закономерности для доказательства сложных теорем, открывает новые возможности для познания законов функционирования сознания.

7 СИСТЕМНЫЕ ПРОГРАММЫ ВСПОМОГАТЕЛЬНОГО НАЗНАЧЕНИЯ

Работая на персональном компьютере с прикладными программами, нельзя ограничиваться только умением управлять этими программами. Практическая работа и реальные ситуации в компьютере часто требуют более широких знаний и, прежде всего, умения сохранять в целостности информацию, защищать ее от возможных разрушений, вызванных дефектами магнитных дисков, сбоями в работе компьютера, воздействиями программных вирусов или ошибками пользователя. И здесь на помощь пользователю могут прийти различные программы вспомогательного назначения (утилиты), относящиеся к группе системных программ. Таких программ разработано очень много. Некоторые из них объединяются в специальные программные комплексы (пакеты программ). Самый известный из таких программных пакетов – Norton Utilities. Рассмотрим некоторые наиболее распространенные группы программ-утилит.

ПРОГРАММЫ-АРХИВАТОРЫ

Одним из наиболее широко распространенных видов сервисных программ являются программы, предназначенные для архивации, упаковки файлов путем сжатия хранимой в них информации.

Сжатие информации – это процесс преобразования информации, хранящейся в файле, к виду, при котором уменьшается избыточность в ее представлении и соответственно требуется меньший объем памяти для хранения.

Сжатие информации в файлах производится за счет устранения избыточности различными способами, например, за счет упрощения кодов, исключения из них постоянных битов или представления повторяющихся символов или повторяющейся последовательности символов в виде коэффициента повторения и соответствующих символов. Применяются различные алгоритмы подобного сжатия информации.

Сжиматься могут как один, так и несколько файлов, которые в сжатом виде помещаются в так называемый архивный файл или архив.

Архивный файл – это специальным образом организованный файл, содержащий в себе один или несколько файлов в сжатом виде и служебную информацию об именах файлов, дате и времени их создания или модификации, размерах и т.п.

Целью упаковки файлов обычно являются обеспечение более компактного размещения информации на диске, сокращение времени и соответственно стоимости передачи информации по каналам связи в компьютерных сетях. Кроме того, упаковка в один архивный файл группы файлов существенно упрощает их перенос с одного компьютера на другой, сокращает время копирования файлов на диски, позволяет защитить информацию от несанкционированного доступа, способствует защите от заражения компьютерными вирусами.

Степень сжатия файлов характеризуется, как правило, коэффициентом K_c, определяемым как отношение объема сжатого файла V_c к объему исходного файла V_0 выраженное в процентах

$$K_{\text{с}} = \frac{V_{\text{с}}}{V_0} \cdot 100\% .$$

Степень сжатия зависит от используемой программы, метода сжатия и типа исходного файла. Программы для архивации отличаются используемыми методами сжатия, что соответственно влияет на степень сжатия.

Программы, осуществляющие работу с архивными файлами, их упаковку и распаковку называются *программами-архиваторами.*

Архивация (упаковка) – помещение (загрузка) исходных файлов в архивный файл в сжатом или несжатом виде.

Разархивация (распаковка) – процесс восстановления файлов из архива точно в таком виде, какой они имели до загрузки в архив. При распаковке файлы извлекаются из архива и помещаются на диск или в оперативную память.

Большие по объему архивные файлы могут быть размещены на нескольких дисках (томах). Такие архивы называются *многотомными.* Том – это составная часть многотомного архива. Создавая архив из нескольких частей, можно записать его части на несколько дискет.

Существует несколько десятков различных программ-архиваторов, которые отличаются перечнем функций и параметрами работы, однако лучшие из них имеют примерно одинаковые характеристики. Из числа наиболее популярных программ можно выделить программы-архиваторы, работающие с ZIP-архивами, RAR-архивами и ARJ-архивами. Обычно упаковка и распаковка файлов выполняются одной и той же программой, но в некоторых случаях это осуществляется разными программами, например, работа с ZIP-архивами под управлением MS DOS: программа PKZIP производит упаковку файлов, а PKUNZIP – распаковку файлов.

Программы-архиваторы позволяют создавать и такие архивы, для извлечения из которых содержащихся в них файлов не требуются какие-либо программы-архиваторы, так как сами архивные файлы могут содержать программу распаковки. Такие архивные файлы называются самораспаковывающимися.

Самораспаковывающийся архивный файл – это загрузочный, исполняемый модуль, который способен к самостоятельной разархивации находящихся в нем файлов без использования программы-архиватора.

Самораспаковывающийся архив получил название SFX-архив (SelF-eXtracting). Архивы такого типа обычно создаются в форме файла с расширением EXE.

Управление программой-архиватором осуществляется одним из двух способов:

- с помощью командной строки MS DOS, в которой формируется команда запуска, содержащая имя программы-архиватора, команду управления и ключи ее настройки, а также имена архивного и исходного файлов; подобное управление характерно для большинства архиваторов, работающих под управлением ОС MS DOS: ARJ, ZIP, PAK, LHA и других;
- с помощью встроенной оболочки и диалоговых панелей, появляющихся после запуска программы и позволяющих вести управление с использованием меню и функциональных клавиш, что создает для пользователя более комфортные условия работы. Такое управление имеет программа-архиватор RAR, работающая под управлением ОС MS DOS, и все архиваторы, работающие под управлением ОС Windows: WinZIP, WinRAR, WinARJ.

Выполняя предписанные ей действия, программа-архиватор, как правило, выводит на экран протокол своей работы.

Все программы-архиваторы оснащены экранами помощи, которые для DOS-программ вызываются при вводе в командной строке только одного имени программы или имени с ключом /?. Помощь может быть краткой – на одном экране или развернутой – на нескольких. Многие архиваторы имеют экраны помощи с примерами составления команд для выполнения различных операций.

ПРОГРАММЫ ОБСЛУЖИВАНИЯ МАГНИТНЫХ ДИСКОВ

В настоящее время магнитные диски персонального компьютера (ПК) являются основными носителями информации, предназначенными для длительного и надежного ее хранения. В процессе работы ПК непрерывно происходит обмен информацией между дисками и оперативной памятью, при этом наиболее интенсивно происходит обмен с жестким диском. Несмотря на высокое качество изготовления дисков и дисковых устройств, в практике регулярной работы на компьютере нередко возникают ситуации, когда не удается прочитать информацию с дисков, происходят нарушения в работе файловой системы, значительно сокращается свободное пространство на дисках или диски оказываются переполненными. Нередко ошибочно удаляются нужные файлы.

Эти нарушения в работе дисков могут возникать по следующим причинам:

- физическое повреждение диска;
- загрязнение магнитной поверхности диска;
- аварийное отключение компьютера;
- несвоевременное извлечение дискет из дисководов;
- перезагрузка операционной системы после аварийного завершения задания;
- воздействие программных вирусов.

Кроме того, при интенсивной эксплуатации компьютера на дисках, главным образом на жестких, накапливаются такие изменения в расположении файлов, которые, если не принимать мер, могут привести к существенному замедлению обмена с ними информацией.

Рассмотрим некоторые программные средства для восстановления удаленных файлов, ремонта поврежденной файловой системы, восстановления информации поврежденных файлов и каталогов, профилактики файловой системы с целью ускорения обмена с жестким диском.

Восстановление поврежденных файлов и дисков

При эксплуатации магнитных дисков нередко приходится сталкиваться с нарушениями в их работе, вызванными логическими или физическими дефектами, в результате которых информация не может быть правильно считана или записана.

Физические дефекты возникают главным образом из-за механических повреждений, воздействия электромагнитных полей или старения магнитного покрытия диска. Наличие физического дефекта делает непригодными к использованию некоторые сектора и кластеры. Если оказывается, что какой-либо файл располагается в таких дефектных секторах или кластерах, то полностью спасти находящуюся в файле информацию не представляется возможным. Дефектные сектора диска должны быть исключены из дальнейшего использования.

Логические дефекты диска связаны с повреждениями системной области диска, включающей таблицу разделов диска (жесткого), таблицу размещения файлов (FAT), загрузочный сектор, каталоги диска. Такие нарушения могут возникать при аварийном отключении питания, сбоях, зависании ошибочно работающих программ, воздействии компьютерных вирусов и других причинах. Во всех этих случаях оказываются незавершенными процедуры работы с дисками, файлами или каталогом. В результате на диске образуются потерянные кластеры, которые или не принадлежат ни одному файлу, или, наоборот, принадлежат сразу нескольким.

Логические дефекты приводят к разрушению файловой системы, "засорению" дискового пространства, когда кластеры считаются занятыми, но не принадлежат ни одному из файлов, а некоторые файлы оказываются "связанными" друг с другом общими кластерами.

Для диагностики дисков и восстановления поврежденных файлов и дисков используются специально разработанные программы, такие как CHKDSK, SCANDISK, входящие в состав операционных систем MS DOS и Windows, и программа NDD (Norton Disk Doctor), входящая в состав программ Norton Utilities.

Устранение фрагментации дисков и ускорение их работы

Операционная система записывает файлы на диск физическими блоками, называемыми кластерами. Кластер занимает один или несколько смежных секторов и может иметь размер в зависимости от емкости диска и версии операционной системы от 2 до 32 Кбайт для жестких дисков и 512 байт для гибких дисков.

После форматирования диска или в том случае, когда на нем имеется достаточно свободного пространства, операционная система записывает файл в смежные, примыкающие друг к другу кластеры. Считывание информации из такого файла происходит при минимальном перемещении магнитных головок. По мере записи на диск новых файлов свободное пространство на нем уменьшается.

При недостаточном размере непрерывного свободного пространства на диске операционная система использует для размещения нового файла имеющиеся свободные участки, помещая в них отдельные цепочки кластеров файла – фрагменты файла. При этом фрагменты могут располагаться на значительном расстоянии друг от друга, что приводит к существенному увеличению времени на перемещение магнитных головок и соответственно времени считывания или записи файла.

Файл, который занимает на диске более одного непрерывного участка, называется *фрагментированным*.

Фрагментация диска – это появление на диске множества свободных участков, разделенных занятыми участками.

Для увеличения свободного пространства диска производят его чистку, т.е. удаление неиспользуемых файлов. Однако удаление файлов еще больше способствует фрагментации, так как освободившиеся участки будут использоваться операционной системой для размещения фрагментов новых файлов. При этом может возникнуть ситуация, при которой свободного пространства на диске много, но все оно состоит из множества разбросанных по диску мелких участков, недостаточных для размещения файлов целиком.

Если в процессе длительной эксплуатации диска, особенно жесткого, не принимать специальных мер, то фрагментированной окажется большая часть файлов и это может замедлить работу диска и соответственно программ, взаимодействующих с ним, в несколько раз. Кроме того, наличие фрагментации всегда ухудшает прогноз восстановления удаленных файлов и каталогов.

В настоящее время разработаны и широко применяются специальные программы, устраняющие фрагментацию дисков. К наиболее распространенным можно отнести программу-утилиту SpeedDisk из пакета сервисных программ Norton Utilities и программа DEFRAG, входящая в состав операционных систем MS DOS и Windows.

Восстановление информации на дисках

В процессе работы на компьютере нередки случаи ошибочного или случайного удаления файлов и каталогов, содержащих ценную для пользователя информацию. Конечно, если сохранились точные копии удаленных файлов, то особых проблем не возникает. Но, если копий нет, то восстановить удаленную информацию могут специально разработанные программы:

- *UnDelete*, входящая в состав операционной системы MS DOS;
- *UnErase*, входящая в пакет сервисных программ Norton Utilities.

Восстановление удаленных файлов и каталогов – это воссоздание на диске файлов и каталогов точно в таком же виде, какой они имели до удаления.

Для успешного восстановления стертых файлов и каталогов очень важно, чтобы после удаления не производилась запись какой-либо информации на тот диск на котором находились эти файлы и, чтобы файлы не были фрагментированы.

При удалении файла или каталога операционная система MS DOS не стирает сами данные с диска, а заменяет в каталоге код первого символа в соответствующем имени на 229, соответствующий в альтернативной таблице кодировки русской строчной букве "х". Вся остальная информация о файле в каталоге, включая номер первого занятого файлом кластера, остается без изменения. При последующих просмотрах каталога этот символ воспринимается операционной системой как сигнал о том, что данное место в каталоге пусто и в него можно заносить информацию о новом файле.

Таким образом, для того чтобы восстановить файл, программа должна найти и восстановить запись о файле в каталоге. Дальнейшие действия зависят от размера и расположения файла. Если файл занимает:

- один кластер, то утилита отыскивает в записи каталога номер начального кластера, обращается к FAT и назначает этот кластер файлу;
- несколько соседних кластеров, то утилита отыскивает в записи каталога номер начального кластера, обращается к FAT и назначает столько последовательно расположенных свободных кластеров, начиная с начального, сколько требует файл;
- несколько непоследовательно расположенных кластеров, т.е. фрагментирован, то при автоматическом восстановлении в FAT утилита назначает требуемое число близлежащих свободных кластеров, а при ручном восстановлении утилита выводит свободные кластер за кластером, и пользователь решает, включать или не включать данный кластер в восстанавливаемый файл.

Таким образом, условиями успешного и полного восстановления удаленного файла (каталога) являются:

- непрерывность расположения файла (каталога);
- незанятость используемых ранее файлом (каталогом) кластеров вновь созданными файлами или каталогами.

КОМПЬЮТЕРНЫЕ ВИРУСЫ И АНТИВИРУСНЫЕ ПРОГРАММНЫЕ СРЕДСТВА

Массовое применение ПК, к сожалению, спровоцировало появление самовоспроизводящихся программ-вирусов, препятствующих нормальной работе компьютера, разрушающих файловую структуру дисков и наносящих ущерб хранимой в компьютере информации. Проникнув в один компьютер, компьютерный вирус способен распространиться на другие компьютеры.

Компьютерным вирусом называется специально написанная программа, способная самопроизвольно присоединяться к другим программам, создавать свои копии и внедрять их в файлы, системные области компьютера и в вычислительные сети с целью нарушения работы программ, порчи файлов и каталогов, создания всевозможных помех в работе на компьютере.

Несмотря на принятые во многих странах законы о борьбе с компьютерными преступлениями и разработку специальных программных средств защиты от вирусов, количество новых программных вирусов постоянно растет. Это требует от пользователя ПК знаний о природе вирусов, способах заражения вирусами и защиты от них.

Основными путями проникновения вирусов в компьютер являются съемные диски (гибкие и лазерные), а также компьютерные сети. Заражение жесткого диска вирусами может произойти при загрузке компьютера с дискеты, содержащей вирус. Такое заражение может быть и случайным, например, если дискету не вынули из дисковода A: и перезагрузили компьютер, при этом дискета может и не быть системной. Заразить дискету гораздо проще. На нее вирус может попасть даже, если дискету просто вставили в дисковод зараженного компьютера и, например, прочитали ее оглавление.

Зараженный диск – это диск в загрузочном секторе которого находится программа-вирус.

Теперь рассмотрим для примера вирус. В загрузочных вирусах выделяют две его части: так называемые *голову* и *хвост*. Заражая дискету, вирус производит следующие действия:

- выделяет некоторую область диска и помечает ее как недоступную операционной системе, что можно сделать по-разному, в простейшем и традиционном случае занятые вирусом секторы помечаются как сбойные;
- копирует в выделенную область диска свой хвост и оригинальный (здоровый) загрузочный сектор;
- замещает программу начальной загрузки в загрузочном секторе своей головой;
- организует новую цепочку передачи управления по схеме: голова вируса – хвост вируса – программа начальной загрузки.

Таким образом, голова вируса теперь первой получает управление, вирус устанавливается в память и передает управление оригинальному загрузочному сектору.

После запуска программы, содержащей вирус, становится возможным заражение других файлов. Наиболее часто вирусом заражаются загрузочный сектор диска и исполняемые файлы, имеющие расширения EXE, COM, SYS или BAT.

Зараженная программа – это программа, содержащая внедренную в нее программу-вирус.

При заражении компьютера вирусом очень важно своевременно его обнаружить. Для этого следует знать об основных признаках проявления вирусов. Эти признаки могут быть следующими:

- прекращение работы или неправильная работа ранее успешно функционировавших программ;
- медленная работа компьютера;
- невозможность загрузки операционной системы;
- исчезновение файлов и каталогов или искажение их содержимого;
- изменение даты и времени модификации файлов;
- изменение размеров файлов;
- неожиданное значительное увеличение количества файлов на диске;
- существенное уменьшение размера свободной оперативной памяти;
- вывод на экран непредусмотренных сообщений или изображений;
- подача непредусмотренных звуковых сигналов;

- частые зависания и сбои в работе компьютера.

Следует заметить, что вышеперечисленные явления не обязательно вызываются присутствием вируса, а могут быть следствием других причин. Поэтому правильная диагностика состояния компьютера всегда затруднительна.

Все существующие в настоящее время вирусы можно классифицировать по среде обитания; способу заражения среды обитания; воздействию; особенностям алгоритма.

В зависимости от среды обитания вирусы можно разделить на сетевые, файловые, загрузочные и файлово-загрузочные. *Сетевые вирусы* распространяются по различным компьютерным сетям. *Файловые вирусы* внедряются главным образом в исполняемые модули, т.е. в файлы, имеющие расширения 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. *Загрузочные вирусы* внедряются в загрузочный сектор диска (Boot-сектор) или в сектор, содержащий программу загрузки системного диска (Master Boot Record). *Файлово-загрузочные вирусы* заражают как файлы, так и загрузочные сектора дисков.

По способу заражения вирусы делятся на резидентные и нерезидентные. *Резидентный вирус* при заражении (инфицировании) компьютера оставляет в оперативной памяти свою резидентную часть, которая потом перехватывает обращение операционной системы к объектам заражения (файлам, загрузочным секторам дисков и т.п.) и внедряется в них. Резидентные вирусы находятся в памяти и являются активными вплоть до выключения или перезагрузки компьютера. *Нерезидентные вирусы* не заражают память компьютера и являются активными ограниченное время.

По степени воздействия вирусы бывают неопасные, опасные и очень опасные. *Неопасные вирусы,* не мешают работе компьютера, но уменьшают объем свободной оперативной памяти и памяти на дисках. Действия таких вирусов проявляются в каких-либо графических или звуковых эффектах. *Опасные* вирусы могут привести к различным нарушениям в работе компьютера. *Очень опасные вирусы* – вирусы, воздействие которых может привести к потере программ, уничтожению данных, стиранию информации в системных областях диска.

По особенностям алгоритма вирусы трудно классифицировать из-за большого разнообразия. Простейшие вирусы – *паразитические,* они изменяют содержимое файлов и секторов диска и могут быть достаточно легко обнаружены и уничтожены. Можно отметить *вирусы-репликаторы,* называемые *червями,* которые распространяются по компьютерным сетям, вычисляют адреса сетевых компьютеров и записывают по этим адресам свои копии. Известны *вирусы-невидимки,* называемые *стелс-вирусами,* которые очень трудно обнаружить и обезвредить, так как они перехватывают обращения операционной системы к пораженным файлам и секторам дисков и подставляют вместо своего тела незараженные участки диска. Наиболее трудно обнаружить *вирусы-мутанты*, содержащие алгоритмы шифровки-расшифровки, благодаря которым копии одного и того же вируса не имеют ни одной повторяющейся цепочки байтов. Имеются и так называемые квазивирусные или *"троянские"* программы, которые хотя и не способны к самораспространению, но очень опасны, так как, маскируясь под полезную программу, разрушают загрузочный сектор и файловую систему дисков. В последние годы появился класс так называемых *макро-вирусов*. Они содержат скрытые команды для некоторых оболочек Windows, которые нежелательны для рядового пользователя. И этот код уже не является кодом для компьютера, то есть это уже не программа, а текст программы, выполняемый оболочкой. Таким образом, он может быть записан в любом необходимом формате: html, htm – для Internet Explorer, doc, xls, xlw, txt, prt, или любой другой – для Microsoft Office и т.д. Такие вирусы наносят вред только определенного характера, ведь оболочка не имеет команд, к примеру, для форматирования жесткого диска. Но все же этот вид вирусов заслуживает внимания, ведь с помощью скрытых гиперссылок он способен самостоятельно загрузить из Internet на ваш компьютер тело вируса, а некоторые вирусы способны обновляться и загружаться по частям через Internet с определенных серверов.

АНТИВИРУСНЫЕ ПРОГРАММЫ

Для обнаружения, удаления и защиты от компьютерных вирусов разработано несколько видов специальных программ, которые позволяют обнаруживать и уничтожать вирусы. Такие программы называются *антивирусными.*

Различают следующие виды антивирусных программ:

- программы-детекторы;

- программы-доктора или фаги;
- программы-ревизоры;
- программы-фильтры;
- программы-вакцины или иммунизаторы.

Программы-детекторы осуществляют поиск характерной для конкретного вируса последовательности байтов (сигнатуры вируса) в оперативной памяти и в файлах, при обнаружении выдают соответствующее сообщение. Недостатком таких антивирусных программ является то, что они могут находить только те вирусы, которые известны разработчикам таких программ.

Программы-доктора или *фаги,* а также *программы-вакцины* не только находят зараженные вирусами файлы, но и "лечат" их, т.е. удаляют из файла тело программы вируса, возвращая файлы в исходное состояние. В начале своей работы фаги ищут вирусы в оперативной памяти, уничтожая их, и только затем переходят к "лечению" файлов. Среди фагов выделяют *полифаги,* т.е. программы-доктора, предназначенные для поиска и уничтожения большого количества вирусов. Наиболее известными полифагами являются программы *Norton AntiVirus* и *DrWeb*, *AVP Касперского.*

Учитывая, что постоянно появляются новые вирусы, программы-детекторы и программы-доктора быстро устаревают, и требуется регулярное обновление их версий.

Программы-ревизоры относятся к самым надежным средствам защиты от вирусов. Ревизоры запоминают исходное состояние программ, каталогов и системных областей диска тогда, когда компьютер не заражен вирусом, а затем периодически или по желанию пользователя сравнивают текущее состояние с исходным. Обнаруженные изменения выводятся на экран видеомонитора. Как правило, сравнение состояний производят сразу после загрузки операционной системы. При сравнении проверяются длина файла, код циклического контроля (контрольная сумма файла), дата и время модификации, другие параметры. Программы-ревизоры имеют достаточно развитые алгоритмы, обнаруживают стелс-вирусы и могут даже отличить изменения версии проверяемой программы от изменений, внесенных вирусом. К числу программ-ревизоров относится широко распространенная в России программа *ADInf* фирмы "Диалог-Наука".

Программы-фильтры или *"сторожа"* представляют собой небольшие резидентные программы, предназначенные для обнаружения подозрительных действий при работе компьютера, характерных для вирусов. Такими действиями могут являться:

- попытки коррекции файлов с расширениями COM и EXE;
- изменение атрибутов файлов;
- прямая запись на диск по абсолютному адресу;
- запись в загрузочные сектора диска;
- загрузка резидентной программы.

При попытке какой-либо программы произвести указанные действия "сторож" посылает пользователю сообщение и предлагает запретить или разрешить соответствующее действие. Программы-фильтры весьма полезны, так как способны обнаружить вирус на самой ранней стадии его существования до размножения. Однако они не "лечат" файлы и диски. Для уничтожения вирусов требуется применить другие программы, например фаги. К недостаткам программ-сторожей можно отнести их "назойливость" (например, они постоянно выдают предупреждение о любой попытке копирования исполняемого файла), а также возможные конфликты с другим программным обеспечением. Примером программы-фильтра является программа *SpIDer Guard,* входящая в состав пакета DrWeb.

Вакцины или *иммунизаторы* — это резидентные программы, предотвращающие заражение файлов. Вакцины применяют, если отсутствуют программы-доктора, "лечащие" этот вирус. Вакцинация возможна только от известных вирусов. Вакцина модифицирует программу или диск таким образом, чтобы это не отражалось на их работе, а вирус будет воспринимать их зараженными и поэтому не внедрится. В настоящее время программы-вакцины имеют ограниченное применение.

Своевременное обнаружение зараженных вирусами файлов и дисков, полное уничтожение обнаруженных вирусов на каждом компьютере позволяют избежать распространения вирусной эпидемии на другие компьютеры.

Основные меры по защите от вирусов

Для того чтобы не подвергнуть компьютер заражению вирусами и обеспечить надежное хранение информации на дисках, необходимо соблюдать следующие правила:

- оснастите свой компьютер современными антивирусными программами и постоянно обновляйте их версии;
- перед считыванием с дискет информации, записанной на других компьютерах, всегда проверяйте эти дискеты на наличие вирусов, запуская антивирусные программы своего компьютера;
- при переносе на свой компьютер файлов в архивированном виде проверяйте их сразу же после разархивации на жестком диске, ограничивая область проверки только вновь записанными файлами;
- периодически проверяйте на наличие вирусов жесткие диски компьютера, запуская антивирусные программы для тестирования файлов, памяти и системных областей дисков с защищенной от записи дискеты, предварительно загрузив операционную систему также с защищенной от записи системной дискеты;
- всегда защищайте свои дискеты от записи при работе на других компьютерах, если на них не будет производиться запись информации;
- обязательно делайте архивные копии ценной для вас информации;
- не оставляйте в дисководе A: дискеты при включении или перезагрузке операционной системы, чтобы исключить заражение компьютера загрузочными вирусами;
- используйте антивирусные программы для входного контроля всех исполняемых файлов, получаемых из компьютерных сетей;
- для обеспечения большей безопасности работы на ПК необходимо использовать программы-ревизоры диска.

8 СЕТЕВЫЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

Эволюция и типы сетей ЭВМ

В 1960-х годах появились первые вычислительные сети (ВС) ЭВМ. По сути, они произвели своего рода техническую революцию, сравнимую с появлением первых ЭВМ, так как была предпринята попытка объединить технологии сбора, хранения, передачи и обработки информации на ЭВМ с техникой связи.

Одной из первых сетей, оказавших влияние на дальнейшее их развитие, явилась сеть ARPA, созданная пятьюдесятью университетами и фирмами США. Она охватывала всю территорию США, часть Европы и Азии. Сеть ARPA доказала техническую возможность и экономическую целесообразность разработки больших сетей для более эффективного использования ЭВМ и программного обеспечения. В настоящее время она преобразовалась в сеть Internet.

В Европе сначала были разработаны и внедрены международные сети EIN и EuroNet, затем появились национальные сети. В 1972 году в Вене была создана сеть MIPSA, в 1979 году к ней присоединились
17 стран Европы, СССР, США, Канада, Япония. Она создана для проведения фундаментальных работ по проблемам энергетики, продовольствия, сельского хозяйства, здравоохранения и т.д. Кроме того, она создала технологию, позволяющую всем национальным институтам развивать связь друг с другом.

В СССР первая сеть разработана в 1960-х годах в Академии наук в Ленинграде. В 1985 году к ней подсоединилась региональная подсеть "Северо-запад" с академическими центрами в Риге и Москве. В 1980 году была сдана в эксплуатацию система телеобработки статистической информации СТОСИ, обслуживавшая Главный вычислительный центр Центрального статистического управления СССР в Москве и республиканские вычислительные центры в союзных республиках.

В настоящее время в мире зарегистрировано более нескольких сотен глобальных сетей.

С появлением микроЭВМ и персональных компьютеров возникли локальные вычислительные сети (ЛВС). Они позволили поднять на качественно новую ступень управление производственными объектами, повысить эффективность использования ЭВМ, поднять качество обрабатываемой информации, реализовать безбумажную технологию, создать новые технологии. Объединение ЛВС и глобальных сетей позволило получить доступ к мировым информационным ресурсам.

ЭВМ, объединенные в сеть, делятся на основные и вспомогательные. Основные ЭВМ — это абонентские ЭВМ (рабочие станции, клиенты). Они выполняют все необходимые информационно-вычислительные работы и определяют ресурсы сети. Вспомогательные ЭВМ (серверы) служат для преобразования и передачи информации от одной ЭВМ к другой по каналам связи и коммутационным машинам (host-ЭВМ). К качеству и мощности серверов предъявляются повышенные требования.

Введем ряд понятий.

Рабочая станция — персональный компьютер, подключенный к сети, через который пользователь получает доступ к ее ресурсам. Рабочая станция сети функционирует как в сетевом, так и в локальном режиме. Она оснащена собственной операционной системой (MS DOS, Windows и т.д.), обеспечивает пользователя всеми необходимыми инструментами для решения прикладных задач.

Клиент — это задача (приложение), рабочая станция или пользователь компьютерной сети, посылающие запрос к серверу. Они отвечают за обработку и вывод информации, а также передачу запросов серверу. ЭВМ клиента может быть любой.

Сервер **—** это персональная или специализированная ЭВМ, выполняющая функции по обслуживанию клиента. Он распределяет ресурсы системы: принтеры, базы данных, программы, внешнюю память и т.д. Существуют сетевые, файловые, терминальные, серверы баз данных, почтовые и др.

Сетевой сервер поддерживает выполнение следующих функций сетевой операционной системы: управление вычислительной сетью, планирование задач, распределение ресурсов, доступ к сетевой файловой системе, защиту информации.

Терминальный сервер поддерживает выполнение функций многопользовательской системы.

Файл-сервер обеспечивает доступ к центральной базе данных удаленным пользователям.

Сервер баз данных **–** многопользовательская система, обеспечивающая обработку запросов клиентов к базам данных. Он является средством решения сетевых задач, в которых локальные сети используются для совместной обработки данных, а не просто для организации коллективного использования удаленных внешних устройств.

Почтовые серверы содержат программы, базы данных, каталоги почтовых ящиков и обеспечивают работу электронной почты.

Host-ЭВМ – ЭВМ, установленная в узлах сети и решающая вопросы коммутации в сети, доступа к сетевым ресурсам: модемам, факс-модемам, серверам и др.

Коммуникационная сеть образуется множеством серверов и host-ЭВМ, соединенных физическими каналами связи, которые называют магистральными. В качестве магистральных каналов выступают телефонные, оптоволоконные кабели, космическая спутниковая связь, беспроводная радиосвязь, медная витая пара.

Классификация вычислительных сетей

В зависимости от территориального расположения абонентских систем вычислительные сети можно разделить на три основных класса:

- глобальные сети (WAN – Wide Area Network);
- региональные сети (MAN – Metropolitan Area Network);
- локальные сети (LAN – Local Area Network).

Глобальная вычислительная сеть объединяет абонентов, расположенных в различных странах, на различных континентах. Взаимодействие между абонентами такой сети может осуществляться на базе телефонных линий связи, радиосвязи и систем спутниковой связи. Глобальные вычислительные сети позволят решить проблему объединения информационных ресурсов всего человечества и организации доступа к этим ресурсам.

Региональная вычислительная сеть связывает абонентов, расположенных на значительном расстоянии друг от друга. Она может включать абонентов внутри большого города, экономического региона, отдельной страны. Обычно расстояние между абонентами региональной вычислительной сети составляет десятки, сотни километров.

Локальная вычислительная сеть (ЛВС) объединяет абонентов, расположенных в пределах небольшой территории. В настоящее время не существует четких ограничений на территориальный разброс абонентов локальной вычислительной сети. Обычно такая сеть привязана к конкретному месту. К классу локальных вычислительных сетей относятся сети отдельных предприятий, фирм, банков, офисов и т.д. Протяженность такой сети можно ограничить пределами 2 – 2,5 км.

Объединение глобальных, региональных и локальных вычислительных сетей позволяет создавать многосетевые иерархии. Они обеспечивают мощные, экономически целесообразные средства обработки огромных информационных массивов и доступ к неограниченным информационным ресурсам. На рис. 6 приведена одна из возможных иерархий вычислительных сетей. Локальные вычислительные сети мо-

гут входить как компоненты в состав региональной сети, региональные сети – объединяться в составе глобальной сети и, наконец, глобальные сети могут также образовывать сложные структуры.

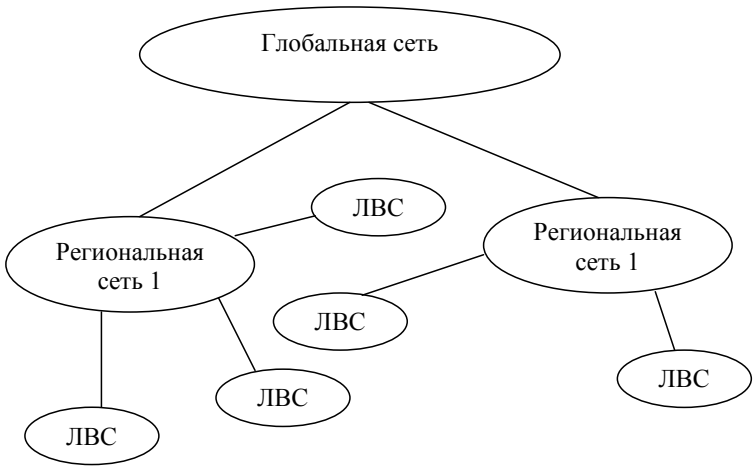


Рис. 6 Иерархия компьютерных сетей

Локальные вычислительные сети

Локальные вычислительные сети за последние годы получили широкое распространение в самых различных областях науки, техники и производства.

Особенно широко ЛВС применяются при разработке коллективных проектов, например, сложных программных комплексов. На базе ЛВС можно создавать системы автоматизированного проектирования. Это позволяет реализовывать новые технологии проектирования изделий машиностроения, радиоэлектроники и вычислительной техники.
В условиях развития рыночной экономики появляется возможность создавать конкурентоспособную продукцию, быстро модернизировать ее, обеспечивая реализацию экономической стратегии предприятия.

В последнее время большой популярностью стали пользоваться *виртуальные локальные сети* VLAN. Их отличие от обычных ЛВС заключается в том, что они не имеют физических ограничений. Виртуальные ЛВС определяют, какие рабочие станции объединяются в конкретные физические группы, что позволяет располагать их в любом месте сети.

Объединение нескольких ЛВС на основе протоколов TCP/IP и HTTP (протокол – правила, по которым взаимодействуют между собой различные компоненты сети) в пределах одного или нескольких зданий одной корпорации получило название *интрасети***.** На принципе интрасети формируются корпоративные сети, подсоединяемые к глобальным сетям. Особое распространение интрасети получили в сети Internet, обеспечивающей так называемую технологию *Intranet*.

Наличие ЛВС в учебных лабораториях университетов позволяет повысить качество обучения и внедрять современные интеллектуальные технологии обучения.

Информационные системы, построенные на базе компьютерных сетей, обеспечивают решение следующих задач: хранение данных, обработка данных, организация доступа пользователей к данным, передача данных и результатов обработки данных пользователям.

В системах централизованной обработки эти функции выполняла центральная ЭВМ (Mainframe, Host-ЭВМ).

Компьютерные сети реализуют распределенную обработку данных. Обработка данных в этом случае распределена между двумя объектами: *клиентом* и *сервером.*

В процессе обработки данных клиент может сформировать запрос на сервер для выполнения сложных процедур, чтение файла, поиск информации в базе данных и т.д. Сервер, определенный ранее, выполняет запрос поступивший от клиента. Результаты выполнения запроса передаются клиенту. Сервер обеспечивает хранение данных общего пользования, организует доступ к этим данным и передает данные клиенту.

Клиент обрабатывает полученные данные и представляет результаты обработки в виде удобном для пользователя. В принципе обработка данных может быть выполнена и на сервере. Для подобных систем приняты термины: системы *клиент-сервер* или архитектура *клиент-сервер.*

Архитектура клиент-сервер может использоваться как в одноранговых локальных вычислительных сетях, так и в централизованной сети с выделенным сервером.

ОДНОРАНГОВАЯ СЕТЬ

В такой сети нет единого центра управления взаимодействием рабочих станций и нет единого устройства для хранения данных. Сетевая операционная система распределена по всем рабочим станциям. Каждая станция сети может выполнять функции как клиента, так и сервера. Она может обслуживать запросы от других рабочих станций и направлять свои запросы на обслуживание в сеть.

Пользователю сети доступны все устройства, подключенные к другим станциям (диски, принтеры).

Достоинства одноранговых сетей: низкая стоимость и высокая надежность.

Недостатки одноранговых сетей:

- зависимость эффективности работы сети от количества станций;
- сложность управления сетью;
- сложность обеспечения защиты информации;
- трудности обновления и изменения программного обеспечения станций.

Структурная схема одноранговой сети изображена на рис. 7.

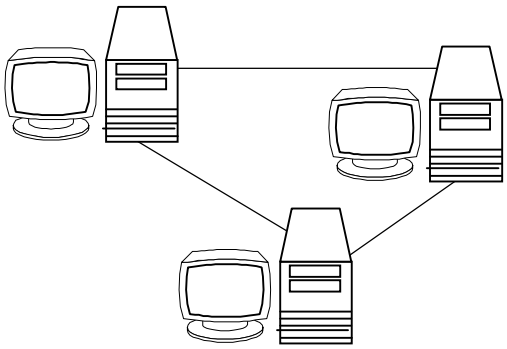


Рис. 7 Одноранговая ЛВС

ЦЕНТРАЛИЗОВАННАЯ СЕТЬ С ВЫДЕЛЕННЫМ СЕРВЕРОМ

В сети с выделенным сервером один из компьютеров выполняет функции хранения данных, предназначенных для использования всеми рабочими станциями, управления взаимодействием между рабочими станциями и ряд сервисных функций.

Такой компьютер обычно называют сервером сети. На нем устанавливается сетевая операционная система, к нему подключаются все разделяемые внешние устройства – жесткие диски, принтеры и модемы.

Взаимодействие между рабочими станциями в сети, как правило, осуществляется через сервер. Роль центрального устройства выполняет сервер. В сетях с централизованным управлением существует возможность обмена информацией между рабочими станциями, минуя файл-сервер.

Достоинства сети с выделенным сервером:

- надежная система защиты информации;
- высокое быстродействие;
- отсутствие ограничений на число рабочих станций;
- простота управления по сравнению с одноранговыми сетями.

Недостатки сети:

- высокая стоимость из-за выделения одного компьютера под сервер;
- зависимость быстродействия и надежности сети от сервера;
- меньшая гибкость по сравнению с одноранговой сетью.

Структурная схема сети с выделенным сервером изображена на рис. 8.

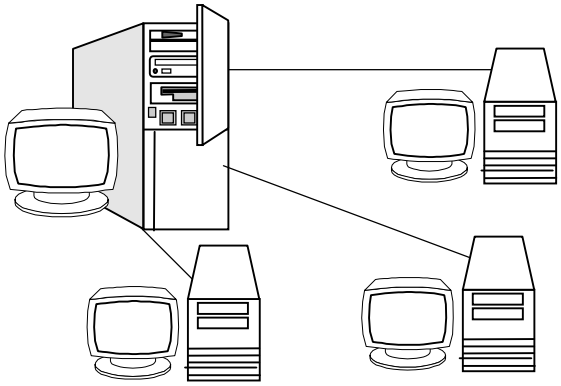


Рис. 8 ЛВС с выделенным сервером

Компоненты локальной вычислительной сети

Основными компонентами ЛВС являются (рис. 9):

1 *Сетевой сервер*, который играет важную роль в управлении ЛВС. Он должен управлять накопителями на жестких дисках и поддерживать коллективные периферийные устройства. Большое значение имеет производительность сервера, при управлении информацией больших объемов и при большом количестве рабочих станций.

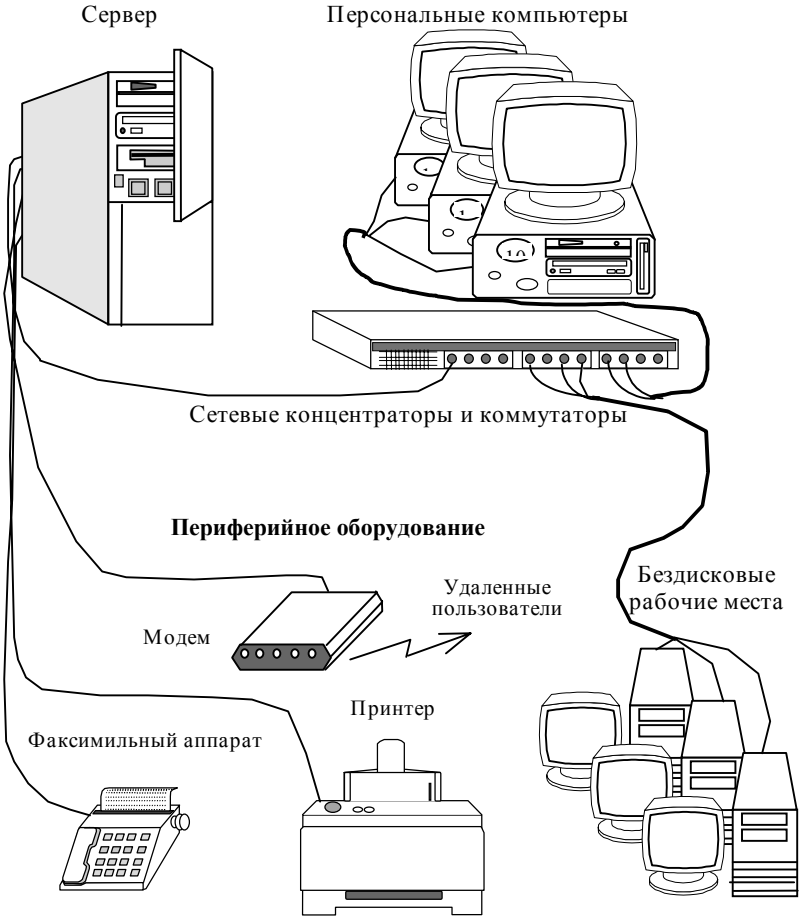


Рис. 9 Компоненты ЛВС

2 *Рабочие места* – автономные компьютерные системы, связанные в сеть и называемые рабочими станциями, автоматизированными рабочими местами и сетевыми станциями. В ЛВС персональный

компьютер используется как рабочее место, располагающее своим собственным процессором с собственным внутренним накопителем и устройством ввода-вывода.

3 При использовании сетевых плат с возможностью автозагрузки можно отказаться от накопителей на гибких магнитных дисках и повысить безопасность хранения данных, так как из этих рабочих станций, часто называемых *PC-терминалами*, не могут быть скопированы данные на транспортные носители данных, а также не могут быть занесены нежелательные данные, например, вирусы.

4 Для подключения большего количества рабочих станций используют сетевые усилители или *коммутаторы*, а также *концентраторы*.

5 *Модем* может связывать удаленных пользователей.

6 Дорогостоящее *периферийное оборудование* (лазерные устройства печати, графопостроители, устройства факсимильной связи, модемы), подключенное к файловому серверу (или другому серверному устройству), можно использовать с любой рабочей станции.

7 *Операционная система*. Как и любая вычислительная система нуждается в программных средствах, объединенных в операционную систему, так и вычислительная сеть нуждается в собственной операционной системе. Всемирно известной операционной системой вычислительных сетей является NetWare фирмы Novell. Функциями операционных систем наделены системы на платформе Windows (Windows 3.11, Windows 95, Windows NT) фирмы Microsoft.

8 *Сетевые адаптеры*. Центральный процессор соединяется с периферийным оборудованием специальным устройством. Для подключения одного ПК к другому требуется устройство сопряжения, которое называется сетевым адаптером или сетевым интерфейсом, модулем, картой. Оно вставляется в свободное гнездо материнской платы. Серверу, в большинстве случаев, необходима сетевая плата с производительностью более высокой, чем у рабочих станций.

4

Характеристики вычислительной сети

Для оценки качества вычислительной сети можно использовать следующие характеристики:

- скорость передачи данных по каналу связи;
- пропускную способность канала связи;
- достоверность передачи информации;
- надежность канала связи и модемов.

Скорость передачи данных по каналу связи измеряется количеством битов информации, передаваемых за единицу времени – секунду. Часто для измерения скорости используется бод – число изменений состояния среды передачи в секунду. Так как каждое изменение состояния может соответствовать нескольким битам данных, то реальная скорость в битах в секунду может превышать скорость в бодах.

Скорость передачи данных зависит от типа и качества канала связи, типа используемых модемов и принятого режима передачи данных.

Для пользователей вычислительных сетей значение имеют не абстрактные биты в секунду, а информация, единицей измерения которой служат байты или знаки. Поэтому более удобной характеристикой канала является его *пропускная способность,* которая оценивается количеством знаков, передаваемых по каналу за единицу времени – секунду. При этом в состав сообщения включаются и все служебные символы. Теоретическая пропускная способность определяется скоростью передачи данных. Реальная пропускная способность зависит от ряда факторов среди которых и способ передачи, и качество канала связи, и условия его эксплуатации, и структура сообщений.

Существенной характеристикой коммуникационной системы любой сети является *достоверность* передаваемой информации. Так как на основе обработки информации о состоянии объекта управления принимаются решения о том или ином ходе процесса, то от достоверности информации, в конечном счете, может зависеть судьба объекта. Достоверность передачи информации оценивают как отношение количества ошибочно переданных знаков к общему числу переданных знаков. Требуемый уровень достоверности должны обеспечивать как аппаратура, так и канал связи. Нецелесообразно использовать дорогостоящую аппаратуру, если относительно уровня достоверности канал связи не обеспечивает необходимых требований.

Для вычислительных сетей этот показатель должен лежать в пределах 10^{-6} … 10^{-7} ошибок/знак, т.е. допускается одна ошибка на миллион переданных знаков или на десять миллионов переданных знаков.

Наконец *надежность* коммуникационной системы определяется либо долей времени исправного состояния в общем времени работы, либо средним временем безотказной работы. Вторая характеристика позволяет более эффективно оценить надежность системы.

Для вычислительных сетей среднее время безотказной работы должно быть достаточно большим и составлять, как минимум, несколько тысяч часов.

Основные топологии ЛВС

Вычислительные машины, входящие в состав ЛВС, могут быть расположены самым случайным образом на территории, где создается вычислительная сеть. Следует заметить, что для управления сетью небезразлично, как расположены абонентские ПК. Поэтому имеет смысл говорить о топологии ЛВС.

Топология ЛВС – это усредненная геометрическая схема соединений ПК в сети.

Топологии вычислительных сетей могут быть самыми различными, но для локальных вычислительных сетей типичными являются всего три: кольцевая, шинная, звездообразная.

Любую компьютерную сеть можно рассматривать как совокупность узлов-устройств, непосредственно соединенных друг с другом.

Кольцевая топология предусматривает соединение узлов сети замкнутой кривой – кабелем передающей среды (рис. 10). Выход одного узла сети соединяется со входом другого. Информация по кольцу передается от узла к узлу. Каждый промежуточный узел между передатчиком и приемником ретранслирует посланное сообщение. Принимающий узел распознает и получает только адресованные ему сообщения.

Кольцевая топология является идеальной для сетей, занимающих сравнительно небольшое пространство. В ней отсутствует центральный узел, что повышает надежность сети. Ретрансляция информации позволяет использовать в качестве передающей среды любые типы кабелей.

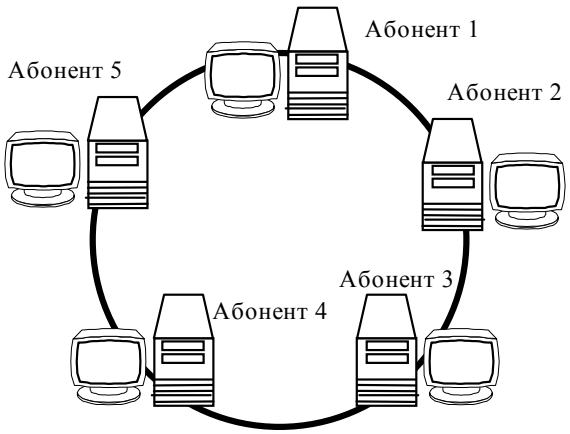


Рис. 10 Сеть кольцевой топологии

Шинная топология – одна из наиболее простых (рис. 11). Она связана с использованием для соединения элементов сети коаксиального кабеля. Данные от передающего узла сети распространяются по шине в обе стороны. Промежуточные узлы не транслируют поступающих сообщений. Информация поступает на все узлы, но принимает сообщение только тот, которому оно адресовано.

Это обеспечивает высокое быстродействие ЛВС с шинной топологией. Сеть легко наращивать и конфигурировать, а также адаптировать к различным системам. Сеть шинной топологии устойчива к возможным неисправностям отдельных узлов. Сети шинной топологии наиболее распространены в настоящее время.

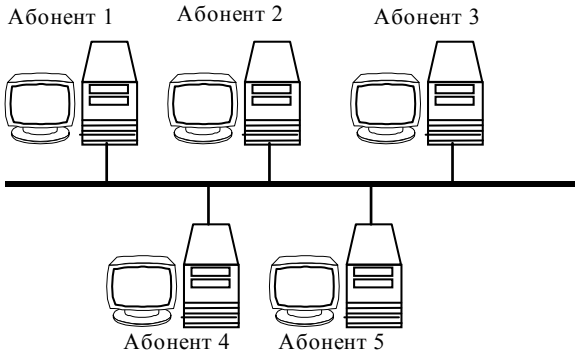


Рис. 11 Сеть шинной топологии

Звездообразная топология (рис. 12) базируется на концепции центрального узла, к которому подключаются периферийные узлы. Каждый периферийный узел имеет свою отдельную линию связи с центральным узлом. Вся информация передается через центральный узел, который ретранслирует, переключает и направляет информационные потоки в сети.

Звездообразная топология значительно упрощает взаимодействие узлов ЛВС друг с другом, позволяет использовать более простые сетевые адаптеры. В то же время работоспособность ЛВС со звездообразной топологией целиком зависит от центрального узла.

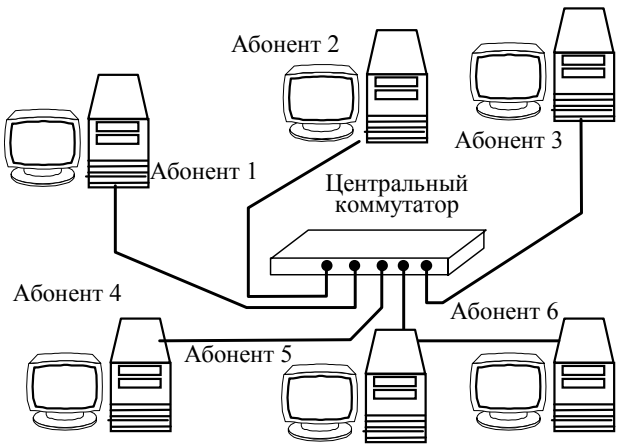


Рис. 12 Сеть звездообразной топологии

В реальных вычислительных сетях могут использоваться более сложные топологии, представляющие в некоторых случаях сочетания рассмотренных.

Выбор той или иной топологии определяется областью применения ЛВС, географическим расположением ее узлов и размерностью сети в целом.

Способы объединения ЛВС

Созданная на определенном этапе развития системы ЛВС с течением времени перестает удовлетворять потребности всех пользователей, и тогда встает проблема расширения ее функциональных возможностей (концентраторы или Hub – устройства, позволяющие развести одну внешнюю линию на несколько внутренних). Может возникнуть необходимость объединения различных ЛВС, появившихся в различных подразделениях в разное время, хотя бы для организации обмена данными с другими системами. Проблема расширения конфигурации сети может быть решена как в пределах ограниченного пространства, так и с выходом во внешнюю среду.

Стремление получить выход на определенные информационные ресурсы может потребовать подключения ЛВС к сетям более высокого уровня.

В самом простом варианте объединение ЛВС необходимо для расширения сети в целом, но если технические возможности существующей сети исчерпаны, то новых абонентов подключить к ней нельзя. Можно только создать еще одну ЛВС и объединить ее с уже существующей.

Для объединения сетей используются следующие устройства:

1 *Мост*. Самый простой вариант объединения ЛВС – объединение одинаковых сетей в пределах ограниченного пространства. Существуют ограничения на длину сетевого кабеля. В пределах допустимой длины строится отрезок сети – сетевой сегмент. Для объединения сетевых сегментов используются *мосты* – устройства, соединяющие две сети, использующие одинаковые методы передачи данных. Для сети персональных компьютеров мост – отдельная ЭВМ со специальным программным обеспечением и дополнительной аппаратурой. Мост может соединять сети разных топологий, но работающие под управлением однотипных сетевых операционных систем.

2 *Маршрутизатор* или *роутер* – устройство, соединяющее сети разного типа, но использующее одну операционную систему. Маршрутизатор выполняет свои функции на сетевом уровне. Маршрутизатор также может выбрать наилучший путь для передачи сообщения абоненту сети, фильтрует информацию, проходящую через него, направляя в одну из сетей только ту информацию, которая ей адресована. Кроме того, маршрутизатор обеспечивает балансировку нагрузки в сети, перенаправляя потоки сообщений по свободным каналам связи.

3 *Шлюз*. Для объединения ЛВС совершенно различных типов, работающих по существенно отличающимся друг от друга протоколам, предусмотрены специальные устройства – шлюзы. *Шлюз* – устройство позволяющее организовать обмен данными между двумя сетями, использующими различные протоколы взаимодействия. С помощью шлюзов можно подключить локальную вычислительную сеть к главному компьютеру, а также локальную сеть подключить к глобальной.

Мосты, маршрутизаторы и даже шлюзы конструктивно выполняются в виде плат, которые устанавливаются в компьютерах. Функции свои они могут выполнять как в режиме полного выделения функций, так и в режиме совмещения их с функциями рабочей станции вычислительной сети.

Физическая передающая среда ЛВС

Физическая среда обеспечивает перенос информации между абонентами вычислительной сети. Физическая передающая среда ЛВС представлена тремя типами кабелей: витая пара проводов, коаксиальный кабель, оптоволоконный кабель.

Витая пара состоит из двух изолированных проводов, свитых между собой (рис. 13). Скручивание проводов уменьшает влияние внешних электромагнитных полей на передаваемые сигналы. Самый простой вариант витой пары – телефонный кабель. Витые пары имеют различные характеристики, определяемые размерами, изоляцией и шагом скручивания. Дешевизна этого вида передающей среды делает ее достаточно популярной для ЛВС.

Рис. 13 Витая пара проводов

Основной недостаток витой пары – плохая помехозащищенность и низкая скорость передачи информации – 0,25 … 1 Мбит/с. Технологические усовершенствования позволяют повысить скорость передачи и помехозащищенность (экранированная витая пара), но при этом возрастает стоимость этого типа передающей среды.

Коаксиальный кабель (рис. 14) по сравнению с витой парой обладает более высокой механической прочностью, помехозащищенностью и обеспечивает скорость передачи информации до 10 … 50 Мбит/с.

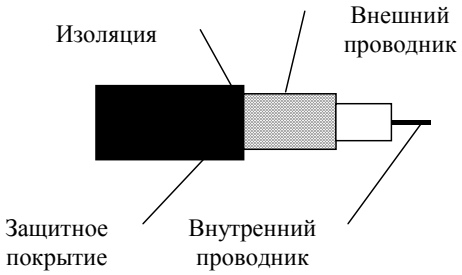


Рис. 14 Коаксиальный кабель

Оптоволоконный кабель – идеальная передающая среда (рис. 15). Он не подвержен действию электромагнитных полей и сам практически не имеет излучения. Последнее свойство позволяет использовать его в сетях, требующих повышенной секретности информации.

Скорость передачи информации по оптоволоконному кабелю более 50 Мбит/с. По сравнению с предыдущими типами передающей среды он более дорог, менее технологичен в эксплуатации.

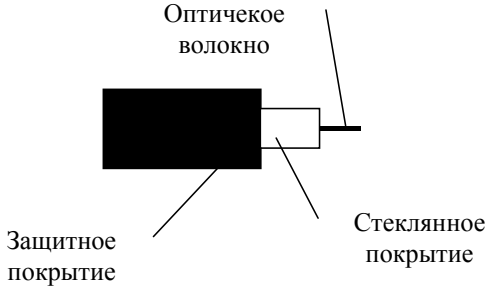


Рис. 15 Оптоволоконный кабель

Глобальная сеть Internet

Internet представляет собой глобальную компьютерную сеть. Само ее название означает "между сетей". Это сеть, соединяющая отдельные сети.

Логическая структура Internet представляет собой некое виртуальное объединение, имеющее свое собственное информационное пространство.

Internet обеспечивает обмен информацией между всеми компьютерами, которые входят в сети, подключенные к ней. Тип компьютера и используемая им операционная система значения не имеют. Соединение сетей обладает громадными возможностями. С собственного компьютера любой абонент Internet может передавать сообщения в другой город, просматривать каталог библиотеки Конгресса в Вашингтоне, знакомиться с картинами на последней выставке в музее Метрополитен в Нью-Йорке, участвовать в видеоконференциях и даже в играх с абонентами сети из разных стран. Internet предоставляет в распоряжение своих пользователей множество всевозможных ресурсов.

Основные ячейки Internet – локальные вычислительные сети. Это значит, что Internet не просто устанавливает связь между отдельными компьютерами, а создает пути соединения для более крупных единиц – групп компьютеров. Если некоторая локальная сеть непосредственно подключена к Internet, то каждая рабочая станция этой сети также может подключаться к Internet. Существуют также компьютеры, самостоятельно подключенные к Internet. Они называются хост-компьютерами (host – хозяин). Каждый подключенный к сети компьютер имеет свой адрес, по которому его может найти абонент из любой точки света.

СИСТЕМА АДРЕСАЦИИ В INTERNET

Сеть Internet можно определить как совокупность ЛВС, удовлетворяющих протоколу TCP/IP, которая имеет общее адресное пространство, где у каждого компьютера есть свой уникальный IP-адрес.

Internet самостоятельно осуществляет передачу данных. К адресам серверов и рабочих станций предъявляются специальные требования. Адрес должен иметь формат, позволяющий вести его обработку автоматически, и должен нести некоторую информацию о своем владельце.

С этой целью для компьютеров, подключенных к Internet, устанавливаются два адреса: цифровой IP-адрес (IP – Internetwork Protocol – межсетевой протокол) для рабочих станций и host-ЭВМ и доменный адрес для host-ЭВМ.

Оба эти адреса могут применяться равноценно. Цифровой адрес удобен для обработки на компьютере, а доменный адрес – для восприятия пользователем.

Цифровой адрес IP-адрес включает в себя три компонента: адрес сети, адрес подсети, адрес компьютера в подсети. Он имеет вид: 192.45.9.200. Адрес сети – 192.45; адрес подсети – 9; адрес компьютера – 200.

Доменный адрес определяет область, представляющую ряд хост-компьютеров. В отличие от цифрового адреса он читается в обратном порядке. Вначале идет имя компьютера, затем имя сети, в которой он находится. Для упрощения связи абонентов Internet все пространство ее адресов разделяется на области – домены. Возможно также разделение по определенным признакам и внутри доменов.

В системе адресов Internet приняты домены, представленные географическими регионами. Они имеют имя, состоящее из двух букв: Франция – fr; Канада – ca; США – us; Россия – ru. Существуют и домены, разделенные по тематическим признакам. Такие домены имеют трехбуквенное сокращенное название: учебные заведения – edu; правительственные учреждения –gov; коммерческие организации – com.

Компьютерное имя включает как минимум два уровня доменов. Каждый уровень отделяется от другого точкой. Слева от домена верхнего уровня располагаются другие имена. Все имена, находящиеся слева, – поддомены для общего домена. Например, существует имя *xxxx.tstu.ru.* Здесь *ru* – общий домен для России; *xxxx* – поддомен *tstu,* который является поддоменом *ru.*

Электронная почта

Самой распространенной стала технология компьютерного способа пересылки и обработки информационных сообщений, позволяющая поддерживать оперативную связь между руководством рабочих групп, сотрудниками, учеными, деловыми людьми, бизнесменами и всеми желающими. Такая технология получила название электронной почты.

Электронная почта (*e-mail – electronic mail*) – специальный пакет программ для хранения и пересылки сообщений между пользователями ЭВМ. Посредством электронной почты реализуется служба безбумажных почтовых отношений. Она является системой сбора, регистрации, обработки и передачи любой информации (текстовых документов, изображений, цифровых данных, звукозаписи и т.д.) по сетям ЭВМ и выполняет такие функции, как редактирование документов перед передачей; их хранение в базе почтового отделения; пересылка корреспонденции; проверка и исправление ошибок, возникающих при передаче; выдача подтверждения о получении корреспонденции адресатом; получение и хранение информации в собственном "почтовом ящике"; просмотр полученной корреспонденции.

Почтовый ящик – специально организованный файл для хранения корреспонденции. Каждый почтовый ящик имеет сетевой адрес IP. Почтовый ящик состоит, как минимум, из двух корзин: отправления и получения.

Для функционирования электронной почты некоторые компьютеры выделяют под *почтовые сервера.* При этом все компьютеры получателей подключены к ближайшему почтовому серверу, получающему, хранящему и пересылающему дальше по сети почтовые отправления, пока они не дойдут до адресата. Почтовые серверы реализуют следующие функции: обеспечение быстрой и качественной доставки информации, управление сеансом связи, проверку достоверности информации и корректировку ошибок, хранение информации "до востребования" и извещение пользователя о поступившей в его адрес корреспонденции, регистрацию и учет корреспонденции, проверку паролей при запросах корреспонденции, поддержку справочников с адресами пользователей.

Пересылка сообщений пользователю может выполняться в индивидуальном, групповом и общем режимах. При *индивидуальном режиме* адресатом является отдельный компьютер пользователя, и корреспонденция содержит его IP адрес. При *групповом режиме* корреспонденция рассылается одновременно группе адресатов. Эта группа может быть сформирована по-разному. Почтовые серверы имеют средства распознавания группы. Например, в качестве адреса может быть указано: "Получить всем, ин-

тересующимся данной темой" или указан список рассылки. В *общем режиме* корреспонденция отправляется всем пользователям – владельцам почтовых ящиков. Посредством двух последних режимов можно организовать телеконференцию, электронные доски объявлений. Во избежание перегрузки почтовых ящиков в почтовых серверах хранятся справочники адресов, содержащих фильтры для групповых и общих сообщений.

Электронная почта поддерживает текстовые процессоры для просмотра и редактирования корреспонденции, информационно-поисковые системы для определения адресата, средства поддержания списка рассылаемой информации, средства предоставления расширенных видов услуг: факс, телекс и т.д.

Электронная почта может быть организована в локальной сети внутри предприятия для обеспечения внутреннего обмена информацией.

Большинство глобальных сетей ЭВМ поддерживает электронную почту. В современных интегрированных пакетах используется объектно-ориентированная технология, а работа пользователя сводится к работе с меню. Почтовый ящик дополняется корзиной для мусора, куда пользователь может поместить ненужную корреспонденцию. Однако в случае необходимости он может оттуда ее забрать или окончательно выбросить.

Формат адреса электронной почты должен иметь вид: *имя пользователя@адрес хост-компьютера.*

Для каждого пользователя на одном хост-компьютере может быть заведен свой каталог для получения сообщений по электронной почте.

- WORLD-WIDE-WEB

(WWW – Всемирная информационная сеть)

Совмещение сетевой технологии с гипертекстом позволило создать новую технологию для работы в сети WWW (World Wide Web). Реализована она на базе сети Internet и получила название "Всемирная паутина". *WWW* является одной из самых популярных информационных служб Internet. Две основные особенности отличают *WWW:* использование гипертекста и возможность клиентов взаимодействовать с другими приложениями Internet.

Гипертекст – текст, содержащий в себе связи с другими текстами, графической, видео- или звуковой информацией. Внутри гипертекстового документа некоторые фрагменты текста четко выделены. Указание на них с помощью, например, мыши позволяет перейти на другую часть этого же документа, на другой документ в этом же компьютере или даже на документы на любом другом компьютере, подключенном к Internet. Связь между гипертекстовыми документами осуществляется с помощью ключевых слов. Найдя ключевое слово, пользователь может перейти в другой документ, чтобы получить дополнительную информацию. Новый документ также будет иметь гипертекстовые ссылки.

Все серверы WWW (Web-серверы, sites) состоят из Web-страниц и используют специальный язык *HTML* (Hypertext Markup Language – язык разметки гипертекста). HTML-документы представляют собой текстовые файлы, в которые встроены специальные команды. Для перемещения по Web-страницам и передачи гипертекстовых документов по сети разработан протокол HTTP (Hyper Text Transfer Protokol). Для поиска Web-страницы с нужным гипертекстовым документом разработаны программы поиска и просмотра, называемые *навигаторами*, или *браузерами* (Brouser) (Netscape Navigator, Internet Explorer). Они обеспечивают интерфейс пользователя со "Всемирной паутиной". При этом стиль оформления экрана и форма представления документа задаются пользователем.

Web-технология заключается в следующем. Пользователь посредством редактора HTML создает гипертекстовый документ. Он размещается на Web-сервере. Администратор делает ссылку в каталоге Web-сервера на Web-страницу, чтобы браузер смог ее найти. Любой другой пользователь посредством браузера может получить доступ к данной Web-странице.

WWW обеспечивает доступ к сети как клиентам, требующим только текстовый режим, так и клиентам, предпочитающим работу в режиме графики.

Работая с Web-сервером, можно выполнить удаленное подключение Telnet, послать абонентам сети электронную почту, получить файлы с помощью FTP–протокола и выполнить ряд других приложений (прикладных программ) Internet. Это дает возможность считать WWW интегральной службой Internet.

ПЕРЕДАЧА ФАЙЛОВ С ПОМОЩЬЮ ПРОТОКОЛА FTP

Назначение электронной почты – прежде всего обмен текстовой информацией между различными компьютерными системами. Не меньший интерес для пользователей сети Internet представляет обмен отдельными файлами и целыми программами.

Для того чтобы обеспечить перемещение данных между различными операционными системами, которые могут встретиться в Internet, используется протокол FTP (File Transfer Protocol), работающий независимо от применяемого оборудования. Протокол обеспечивает способ перемещения файлов между

двумя компьютерами и позволяет абоненту сети Internet получить в свое распоряжение множество файлов. Пользователь получает доступ к различным файлам и программам, хранящимся на компьютерах, подключенных к сети.

Программа, реализующая этот протокол, позволяет установить связь с одним из множества FTP-серверов в Internet.

FTP-сервер – компьютер, на котором содержатся файлы, предназначенные для открытого доступа.

Программа FTP-клиент не только реализует протокол передачи данных, но и поддерживает набор команд, которые используются для просмотра каталога FTP-сервера, поиска файлов и управления перемещением данных.

9 ОСНОВЫ АЛГОРИТМИЗАЦИИ

Основные этапы решения задачи на ЭВМ

В настоящее время на ЭВМ решают самые разнообразные задачи. В каждом случае ЭВМ выполняет какую-то программу, обычно довольно сложную. Некоторые из программ требуют от пользователя специальных знаний и высокой квалификации. Несмотря на бесконечное разнообразие программ, в самом процессе их изготовления можно усмотреть нечто общее и выделить несколько этапов решения задачи на ЭВМ.

1 *Постановка задачи*

Под постановкой задачи понимают математическую или иную строгую формулировку решаемой задачи. Этот этап включает определение целей создаваемой программы и определение ограничений, налагаемых на программу. При постановке задачи должны быть определены требования:

- ко времени решения поставленной задачи;
- объему необходимых ресурсов, например, оперативной памяти;
- точности достигаемого результата.

2 *Проектирование программы*

Если задача вычислительная, то на этом этапе следует выбрать метод расчета, если разрабатывается компьютерная игра, должен быть определен ее сценарий. В любом случае следует выбрать или создать некую формальную модель, которая, в конечном счете, реализуется в будущей программе. На этапе проектирования определяют вид данных, с которыми будет работать программа, основные части, из которых программа будет состоять и характер связей между этими частями.

3 *Разработка алгоритма*

На этом этапе следует разработать детали проекта программы. Детализацию необходимо довести до той степени, когда кодирование деталей программы (перевод их на алгоритмический язык) станет тривиальным. Возможно, детализация потребует нескольких стадий, от крупных блоков ко все более мелким, и в результате должно получиться то, что называется алгоритмом решения задачи.

4 *Написание программы на языке программирования* (*кодирование*)

После того как алгоритм разработан, его записывают на алгоритмическом языке, и этот процесс называют кодированием алгоритма. Для выполнения данного этапа необходимо знать хотя бы один из многих существующих языков программирования, а лучше знать несколько, чтобы выбрать наиболее подходящий для решаемой задачи.

Хотя этап кодирования считается менее творческим, чем предыдущие, для его успешного выполнения требуется хорошее знание как самого языка, так и средств разработки программ: транслятора, компоновщика, программных библиотек и многого другого.

5 *Отладка и тестирование программы*

Целью данного этапа является поиск и устранение ошибок в программе. Ошибки бывают синтаксические (нарушение грамматики алгоритмического языка) и смысловые (искажение самого алгоритма решения задачи). О первых мы не говорим, их обычно обнаруживают и исправляют на этапе кодирования, совершая пробную трансляцию программы. Вторые же можно выявить только в процессе проверки программы на специально подобранных входных данных или в ходе опытной эксплуатации программы.

6 *Получение решения и анализ результатов*

После проверки программы и устранении всех ошибок получают решение поставленной задачи, которое необходимо проанализировать. Если речь идет о моделировании какого-то природного процесса, то следует сравнить полученные с помощью компьютера результаты и результаты наблюдений. Они могут отличаться. В этом случае может потребоваться возврат на один из предыдущих этапов для устранения причин несоответствия результатов.

Разделение процесса разработки программ на шесть этапов носит весьма условный характер. В случае простых программ, некоторые этапы сливаются, например, проектирование с разработкой алгоритма или кодирование с отладкой. В случае сложных программ могут добавиться новые фазы разработки, например, проектирование базы данных.

Более важным является то, что работа над сложной программой состоит в многократном прохождении цикла разработки, так как в процессе тестирования могут быть обнаружены такие ошибки, для исправления которых придется вернуться не только к кодированию или алгоритмизации, но и к проектированию, а в особых случаях – пересмотреть и постановку задачи.

Если же удалось разработать полезную программу, то работа над ней не заканчивается этапом тестирования, а переходит в фазу сопровождения. Программа живет, приобретает новые функции, совершенствует старые, избавляется от последних ошибок и, наконец, умирает, уступив натиску более новых и совершенных программ.

АЛГОРИТМ И ЕГО СВОЙСТВА

Среди перечисленных этапов разработки программ центральное место занимает этап разработки алгоритма.

Само слово "алгоритм" происходит от имени персидского математика Аль Хорезми, который в IX веке разработал правила четырех арифметических действий (сегодня мы бы сказали алгоритмы арифметических действий).

В начале XX века алгоритмы стали объектом изучения математиков, появились различные математические уточнения понятия "алгоритм" и возникла целая отрасль математики – теория алгоритмов. Результаты, полученные теорией алгоритмов, служат теоретическим фундаментом всей компьютерной технологии, но в повседневной программистской практике не используются, поэтому сейчас мы будем обсуждать алгоритмы в их интуитивном, "бытовом" понимании.

Итак, алгоритм – это описание некоторой последовательности действий, приводящее к решению поставленной задачи, но не всякое, а обладающее определенными свойствами. Основными сойствами алгоритма являются:

1) *Дискретность.* Под дискретностью понимается то, что алгоритм состоит из описания последовательности шагов обработки, организованных таким образом, что в начальный момент задаётся исходная ситуация, а после каждого следующего шага ситуация преобразуется на основе данных, полученные в предшествующие шаги обработки. Дискретность алгоритма означает, что он исполняется по шагам: каждое действие, предусмотренное алгоритмом, исполняется только после того, как закончилось исполнение предыдущего, то есть преобразование исходных данных в результат происходит во времени дискретно. Но здесь важно заметить, что не стоит увлекаться детализацией алгоритма и разбиением его на все более и более мелкие шаги.

2) *Детерминированность* (*определенность*). Это свойство означает, что на каждом шаге алгоритма однозначно определяется преобразование данных, полученных на предшествующих шагах алгоритма, то есть на одинаковых исходных данных алгоритм должен всегда давать одинаковые результаты.

3) *Результативность* (*конечность*). Свойство результативности обычно подразумевает завершение работы алгоритма за конечное число шагов (при этом количество шагов может быть заранее не известным и различным для разных исходных данных).

4) *Массовость* (*универсальность*). Это свойство состоит в том, что алгоритм решения задачи разрабатывается в общем виде, то есть он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные выбираются из некоторой области, называемой областью применимости алгоритма.

5) *Понятность.* Каждый шаг алгоритма должен обязательно представлять собой какое-либо допустимое действие, т.е. алгоритм строится для конкретного исполнителя автором и должен быть им обоим понятен. Это облегчает проверку и модификацию алгоритма при необходимости.

СПОСОБЫ ПРЕДСТАВЛЕНИЯ АЛГОРИТМОВ

Алгоритм представляет решение задачи в виде точно определенной последовательности действий по преобразованию исходных данных в результат. Процесс составления алгоритмов называют алгоритмизацией.

Алгоритм, реализующий решение задачи, можно представить различными способами – с помощью графического или текстового описания. Графический способ представления алгоритмов имеет ряд пре-

имуществ благодаря визуальности и явному отображению процесса решения задачи. Алгоритмы, представленные графическими средствами, получили название *блок-схем.*

Текстовое описание алгоритма является достаточно компактным и может быть реализовано на *естественном языке* или *специальном* (*алгоритмическом*) *языке* в виде программы. Таблицы значений представляют алгоритм неявно, как некоторое преобразование конкретных исходных данных в выходные.

Все три способа представления алгоритмов можно считать взаимодополняющими друг друга. На этапе проектирования алгоритмов наилучшим способом является графическое представление, а на этапах проверки и применения алгоритма – текстовая запись в виде программы.

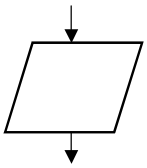
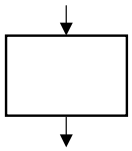
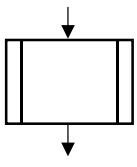
ПРАВИЛА ВЫПОЛНЕНИЯ БЛОК-СХЕМ

Блок-схемой называется наглядное изображение алгоритма, когда отдельные действия (этапы алгоритма) изображаются при помощи различных геометрических фигур (блоков), а связи между этапами (последовательность выполнения этапов) указываются при помощи стрелок, соединяющие эти фигуры.

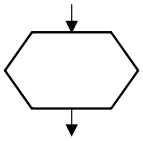
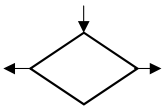
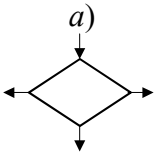
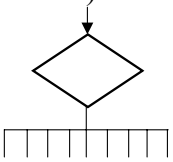
Выполнение блок-схем осуществляется по ГОСТ 19.701–90.

При выполнении блок-схем внутри каждого блока указывается поясняющая информация, которая характеризует действия, выполняемые этим блоком. Наиболее типичные действия алгоритма могут быть изображены в виде блоков, представленных в табл. 1.

1 Графические символы ГОСТ 19.701-90,
используемые в блок-схемах

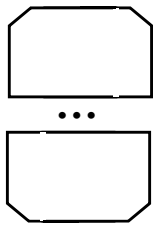
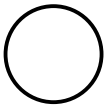
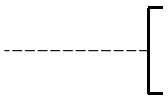
Наименование	Обозначение	Описание
1 Данные		Символ отображает данные, носитель которых не определен. Этот символ используется для обозначения операций ввода данных и вывода результатов, не конкретизируя устройства ввода или вывода. Внутри символа записываются имена данных и производимая над ними операция
2 Процесс		Символ отображает функцию обработки данных любого вида (действие, выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации). Внутри символа указываются выполняемые действия
3 Предопределенный процесс		Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле). Внутри блока записывается имя подпрограммы и параметры, при которых программа будет выполняться

Продолжение табл. 1

Наименование	Обозначение	Описание символа
4 Подготовка		Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы). Внутри символа записывается имя переключателя и условия его модификациии
5 Решение	 *а*)  *б*)  *в*)	Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов (рис. *а*), один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути. В случае, если символ имеет несколько выходов, то их следует показывать: • несколькими линиями от данного символа к другим символам (рис. *б*); • одной линией от данного символа, которая затем разветвляется в соответствующее число линий (рис. *в*). Каждый выход из символа должен сопровождаться соответствующими значениями условий. Внутри символа записывается проверяемое условие

Продолжение табл. 1

Наименование	Обозначение	Описание символа

6 Граница цикла		Символ, состоящий из двух частей, отображает начало и конец цикла. Блоки, составляющие тело цикла записываются между этими символами. Обе части цикла имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т.д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие
7 Соединитель		Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение
8 Терминатор		Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы). Соответствующее действие записывается внутри символа
9 Комментарий		Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснений или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. Текст комментариев или примечаний должен быть помещен около ограничивающей фигуры

Символы в схеме должны быть расположены равномерно. Следует придерживаться разумной длины соединений и минимального числа длинных линий. Формы символов, установленные ГОСТ 19.701–90, должны служить руководством для фактически используемых символов. Не должны изменяться углы и другие параметры, влияющие на соответствующую форму символов. Символы должны быть по возможности одного размера.

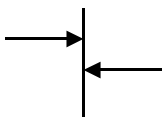


Рис. 16 Объединение двух линий потока

В схемах может использоваться идентификатор символов, который располагается слева над символом (нумерация символов).

Потоки данных в схемах показываются линиями. Направление потока слева направо и сверху вниз считается стандартным. В случаях, когда необходимо внести большую ясность в схему или поток имеет

направление отличное от стандартного, на линиях используются стрелки, указывающие это направление.

В схемах следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменения направления в точках пересечения не допускаются. Если две или более входящих линии объединяются в одну исходящую линию, то место объединения линий смещается (рис. 16).

ОСНОВНЫЕ СТРУКТУРЫ АЛГОРИТМОВ

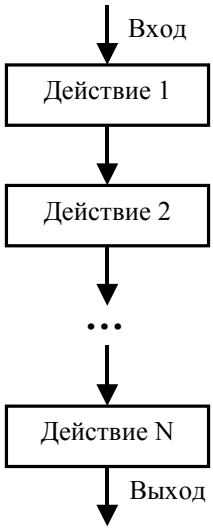


Рис. 17 Линейная структура

Основные структуры алгоритмов (ОСА) – это ограниченный набор блоков и стандартных способов их соединения для выполнения типичных последовательностей действий.

ОСА используются при структурном подходе к разработке алгоритмов и программ, предполагающем использование нескольких основных структур, комбинация которых дает все многообразие алгоритмов и программ.

К основным алгоритмическим структурам относятся линейная, разветвляющиеся и циклические структуры. Рассмотрим их несколько подробнее.

1 *Линейная структура* (*следование*)

Данная структура предполагает последовательное размещение блоков и групп блоков в алгоритмической схеме. В программе линейная структура реализуется последовательным размещением операторов (рис. 17).

2 *Разветвляющиеся структуры*

Эти структуры в своем составе содержат блок условия и различные конструкции ветвления, обеспечивающие альтернативные пути развития вычислительных процессов. Можно выделить несколько разветвляющихся структур, отличающихся друг от друга числом ветвлений и наличием выполняемых действий: разветвление (рис. 18, *а*), обход (рис. 18, *б*), множественный выбор (рис. 18, *в*).

Структура разветвления (рис. 18, *а*) предполагает проверку условия и в зависимости от истинности условия будет выполняться "Действие 1" (если условие принимает значение "истина") или "Действие 2" (если условие принимает значение "ложь").

Структуру обхода (рис. 18, *б*) можно считать частным случаем разветвления, когда в одной из ветвей обработки данных не содержится никаких действий.

Множественный выбор (рис. 18, *в*) предполагает возможность развития вычислительного процесса по одному из нескольких направлений. В процессе решения данной алгоритмической схемы будет выполняться та ветвь обработки данных, для которой вычисленное в блоке "Решение" значение переменной D совпадет со значением одной из констант D_1, D_2, …, D_n.

3 *Циклические структуры*

Циклические алгоритмы являются наиболее распространенным видом алгоритмов, в которых предусматривается повторное выполнение определенного набора действий в зависимости от выполнения некоторого условия.

Существуют два основных вида циклических алгоритмов: циклические алгоритмы с предусловием (цикл "ПОКА"), циклические алгоритмы с постусловием (цикл "ДО"). Они отличаются друг от друга местоположением условия выхода из цикла.

Цикл с предусловием (рис. 19, *а*) начинается с проверки условия выхода из цикла. Если оно ложно, то выполняются те действия, которые составляют тело цикла. В противном случае, если логическое выражение истинно, то этот цикл прекращает свои действия. Особенность этого цикла состоит в том, что цикл может не выполниться ни разу, если условие выхода при первой же проверке даст истинный результат.

Цикл с постусловием (рис. 19, *б*) функционирует иначе. Сначала выполняется один раз те действия, которые составляют тело цикла, затем проверяется логическое выражение, определяющее условие выхода из цикла. Если условие выхода истинно, то цикл с постусловием прекращает свою работу, в противном случае происходит повторение действий, указанных в цикле. Этот цикл всегда выполняется хотя бы один раз, так как первая проверка условия выхода происходит после выполнения действий, составляющих тело цикла.

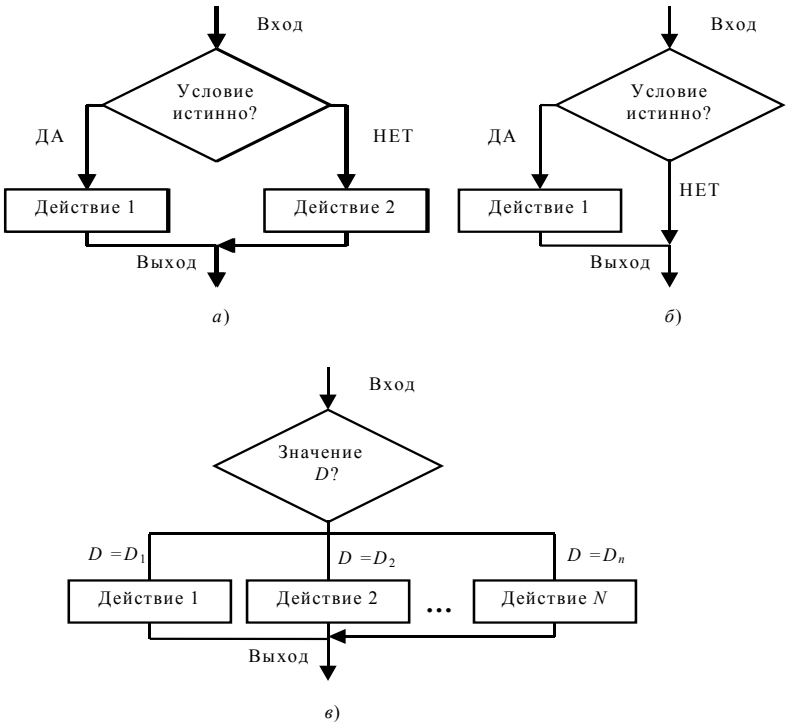


Рис. 18 Разветвляющиеся структуры алгоритмов

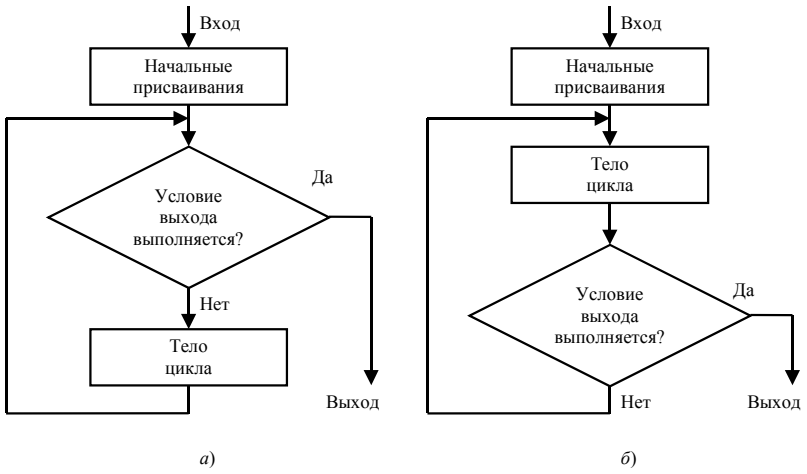


Рис. 19 Циклические структуры алгоритмов

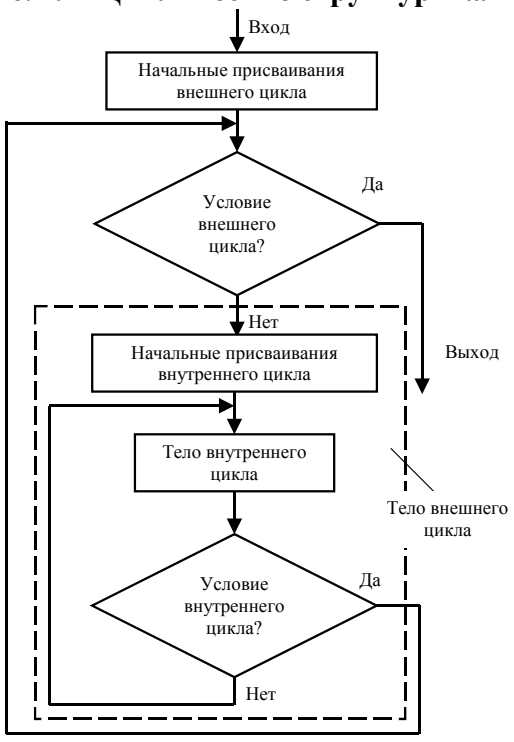


РИС. 20 ВЛОЖЕННАЯ АЛГОРИТМИЧЕСКАЯ СТРУКТУРА "ЦИКЛ В ЦИКЛЕ"

Особенностью рассмотренных структур является то, что все они имеют один вход и один выход и могут быть соединены друг с другом в любой последовательности. В частности, каждая структура может содержать любую другую в качестве одного из блоков. Например, на рис. 20 изображена часто встречающаяся в реальных задачах вложенная циклическая структура ("цикл в цикле"), в которой тело внешнего цикла (внешний цикл – цикл "ПОКА") является также циклом (внутренний цикл – цикл "ДО"). При этом в соответствии с принципами вложенности внутренний цикл заканчивается раньше, чем внешний цикл.

НЕКОТОРЫЕ ПРИЕМЫ АЛГОРИТМИЗАЦИИ

Проектирование алгоритмов — наиболее ответственный этап разработки программ для ЭВМ, который определяет, в конечном итоге, насколько создаваемая программа соответствует предъявляемым к ней требованиям. Затраты на создание, сопровождение и эксплуатацию программных продуктов, научно-технический уровень разработки, время морального устаревания и многое другое — все это также зависит от принятых проектных решений.

Методы проектирования алгоритмов и программ очень разнообразны, но в основе всех их лежат приведенные выше основные структуры алгоритмов и некоторые общие приемы алгоритмизации. Рассмотрим некоторые из таких приемов.

1 *Обмен значениями между двумя переменными*

Замена двух (или более) переменных местами практически всегда осуществляется через третью, дополнительно введенную, переменную. Для этого вначале вновь введенной переменной *C* присваивается значение одной из меняемых переменных, например *A*, а затем последней присваивается значение второй меняемой переменной *B*. И, наконец, последний шаг замены – значение вновь введенной переменной *C* присваивается переменной *B* (рис. 21).

2 *Вычисление произведения (суммы) чисел*

Задача вычисления суммы или произведения значений очень часто встречается при разработке различных алгоритмов. Существуют некоторые общие правила, позволяющие решить эту задачу правильно и быстро. Они заключаются в следующем: вначале необходимо задать некоторое начальное значение для вычисляемой суммы или произведения (как правило, начальное значение суммы принимается равным нулю, а произведения – единице, но в зависимости от конкретной постановки задачи эти начальные значения могут быть другими), а затем в цикле (повторяется столько раз, сколько значений необходимо сложить или перемножить) происходит накапливание данной суммы или произведения. На рис. 22, *а* приведена схема алгоритма вычисления суммы – $\sum_{i=1}^{N} x_i$, а на рис. 22, *б* – схема вычисления факториала от N – $N!$.

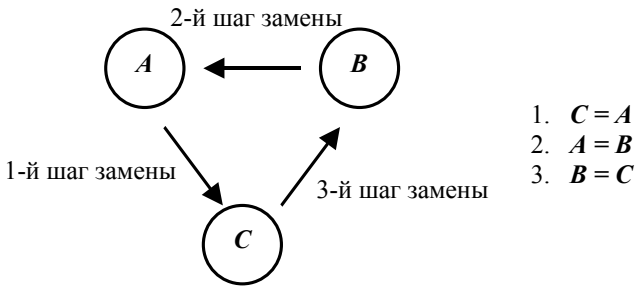
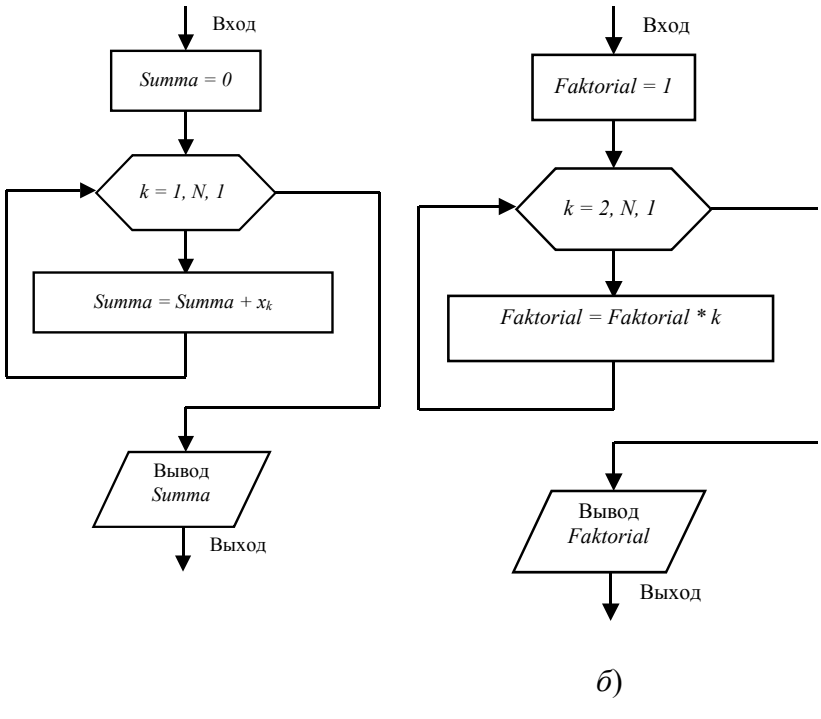


Рис. 21 Схема обмена значениями между двумя переменными

3 *Нахождение наибольшего (наименьшего) значения одномерного массива*

Под структурой данных типа массив понимают однородную структуру однотипных данных, одновременно хранящихся в последовательных ячейках оперативной памяти. Эта структура должна иметь имя и определять заданное количество данных (элементов). Однотипность данных дает возможность использования циклических алгоритмов для обработки всех элементов массива. Количество итераций цикла определяется количеством элементов массива. Одновременное хранение в памяти всех элементов

массива позволяет решать большой набор задач, таких как поиск элементов, упорядочение и изменение порядка следования элементов. Доступ к любому элементу массива осуществляется по его номеру (индексу). Поэтому для обращения к элементу массива используют имя переменной-массива и значение номера элемента.



б)

а)

Рис. 22 Схемы алгоритмов вычисления суммы (*а*) и произведения (*б*)

Рассмотрим алгоритм нахождения наименьшего значения и его индекса в одномерном массиве, состоящем из N элементов – $M(N)$ (рис. 23). При выборе наименьшего значения вначале задают или выбирают некое эталонное значение. Для массива этим значением может являться первый элемент. Затем в цикле начинают перебирать все оставшиеся элементы массива и сравнивать каждый из них с эталонным значением. В случае если текущий элемент оказывается лучше эталона, его принимают за новый эталон. При выборе наибольшего значения поступают аналогичным образом.

4 *Сортировка массивов*

Под сортировкой понимается процесс размещения объектов некоторого списка в порядке следования их значений (по возрастанию или по убыванию). Во всем спектре задач, связанных с применением ЭВМ, сортировке принадлежит исключительно важное место.

Для выполнения операций сортировки существует достаточное количество различных методов, многие из которых, к сожалению, имеют довольно сложные алгоритмы решения. Наиболее простыми и, в то же время, обладающими достаточной эффективностью, являются методы линейной и "пузырьковой" сортировки.

Алгоритм линейной сортировки заключается в следующем. Выбирается первый элемент ряда и последовательно сравнивается со всеми стоящими за ним элементами. В случае если два сравниваемых элемента расположены неверно друг по отношению к другу, то они меняются местами. И так до конца ряда. В итоге на первом месте в ряду окажется либо самый маленький, либо самый большой элемент ряда (в зависимости от сортировки: по убыванию или по возрастанию). Затем берется второй элемент ряда и также сравнивается со всеми стоящими за ним элементами, в случае необходимости выполняется перестановка. Подобные действия выполняются со всеми элементами ряда до конца. После всех описанных действий получается отсортированный ряд. На рис. 24 приведен пример линейной сортировки одномерного массива S(N) по возрастанию.

Несмотря на простоту программной реализации, производительность линейной сортировки невысока, так как для обработки любого n-элементного ряда требуется одно и то же число сравнений. К при-

меру, для сортировки n элементов требуется выполнить около $n^2/2$ сравнений независимо от их первоначального расположения, включая даже случай, когда задан полностью упорядоченный ряд. Таким образом, путь к повышению производительности линейной сортировки лежит через уменьшение общего числа сравнений, для чего необходимо учитывать характер предварительного распределения значений элементов.

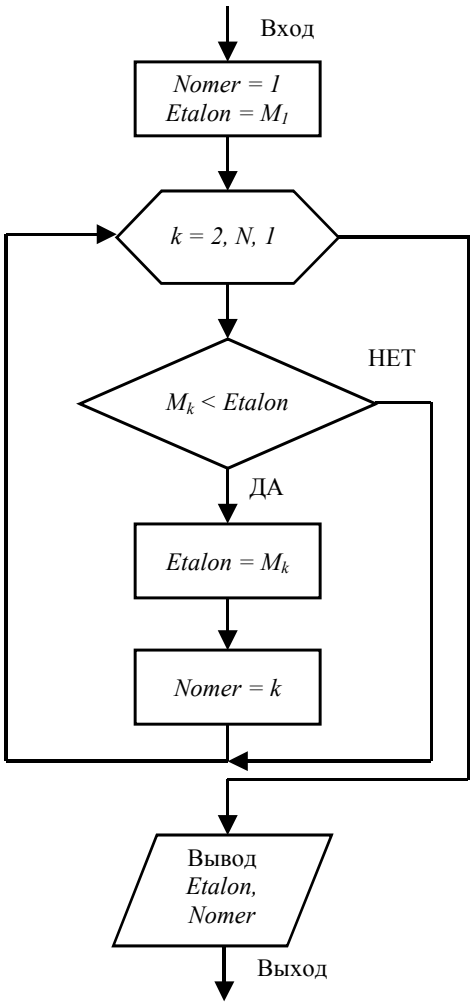


Рис. 23 Нахождение наименьшего значения одномерного массива и его индекса

Указанный недостаток отсутствует в методе "пузырьковой" сортировки, в основе которого лежит следующая идея: если каждый элемент ряда находится в правильном положении по отношению к двум соседним с ним, то весь ряд отсортирован. В соответствии с этой идеей *алгоритм "пузырьковой" сортировки* заключается в следующих действиях. Берется первая пара рядом стоящих элементов (1-й и 2-й элементы) и сравнивается между собой. В случае если элементы стоят неверно друг по отношению к другу, то они меняются местами. Далее таким же образом сравнивается следующая пара (2-й и 3-й элементы) и так до конца ряда, пока не произойдет сравнение предпоследнего и последнего элемента. Такое сравнение всех членов ряда равносильно прохождению одного "пузырька" через ряд. После этого возвращаются в начало ряда и повторяют сравнение всех элементов (второй "пузырек") и т.д. Описанные действия продолжаются многократно – до тех пор, пока очередной проход через ряд не зарегистрирует отсутствие перестановок (для этого можно использовать логическую либо какую другую переменную). В этом случае сортировка считается законченной. На рис. 25 приведен алгоритм "пузырьковой" сортировки одномерного массива по убыванию.

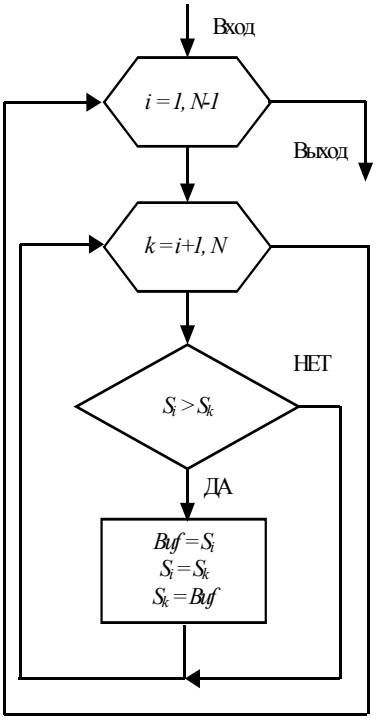


Рис. 24. Линейная сортировка по возрастанию

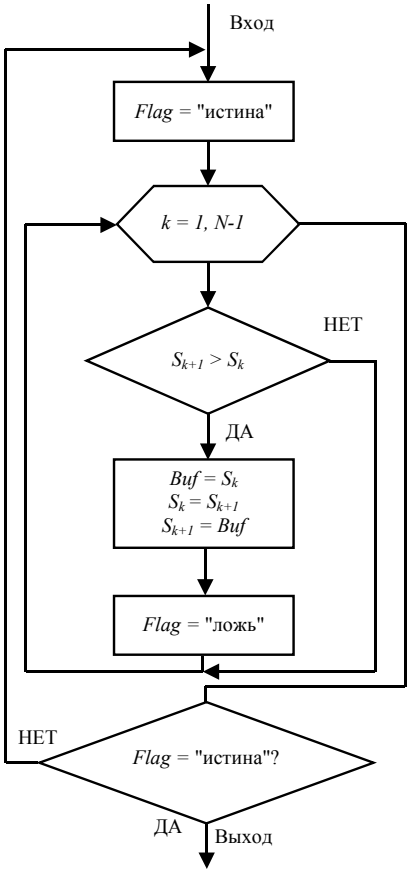


Рис. 25. Пузырьковая сортировка по убыванию

10 ЯЗЫК ПРОГРАММИРОВАНИЯ TURBOPASCAL 7.0. КРАТКИЕ СВЕДЕНИЯ

Изучение языка программирования представляет собой знакомство с формальными правилами записи алгоритмов для их последующего выполнения компьютером. Как и любой алгоритм, являющийся, после- довательностью инструкций (шагов или этапов), программа на языке Turbo Pascal состоит из команд (операторов), записанных в определенном порядке и формате. Эти операторы позволяют получать, сохранять и обрабатывать данные различных типов (числа, символы, строки символов, т.д.) и состоят из "служебных слов" языка. Этих слов не так много, но их значение трудно переоценить. Служебные слова можно ис- пользовать только по своему прямому назначению.

Подробному изложению правил и тонкостей программирования на языке Turbo Pascal посвящены многочисленные издания, учебники и справочники, издаваемые в нашей стране и за рубежом. Поэтому в рамах данного пособия рассмотрим только основные правила записи программ, конструкции и операторы языка Turbo Pascal, представив материал в виде краткой справки.

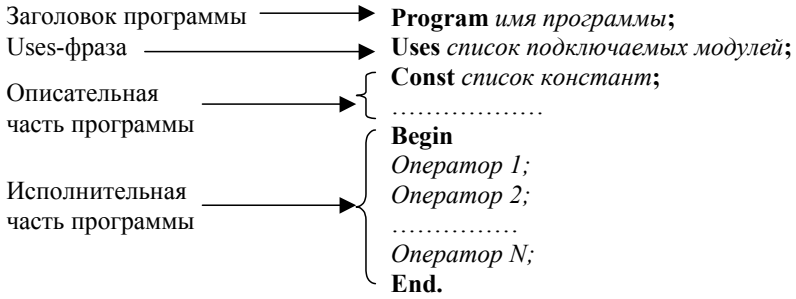
- **Общая структура программ**
-
- Алфавит языка
-

- *Символы используемые в идентификаторах: в*се буквы латинского алфавита, арабские цифры, символ подчеркивания (_). Малые (строчные) и большие (прописные) буквы не различаются. Первым символом может быть только буква или символ подчеркивания. Длина имени может быть от 1 до 127 символов. При этом первые 63 символа в различных именах являются уникальными, т.е. должны обязательно отличаться друг от друга. Под идентификатором понимается имя любого объекта программы.
- *Разделители*: используются для отделения друг от друга идентификаторов, служебных слов и чисел. В качестве разделителей используется пробел или комментарий – любой текст, заключенный

между фигурными скобками **{** и **}** или скобками вида **(*** и ***)**. Текст комментария можно расположить в любой части программы.

- *Специальные символы:* знаки пунктуации (**; : .. [ ] .** и др.), знаки операций (арифметические, логические и др.), зарезервированные слова языка (служебные слова, иена директив и т.п.).
- *Неиспользуемые в конструкциях языка символы:* буквы русского алфавита и некоторые символы (& % и др.).

- *Основные составные части программы*



Заголовок программы является необязательным элементом и может отсутствовать. Uses-фраза используется для подключения к программе внешних модулей. Модуль (Unit) – это набор стандартных операций и описаний, которые можно использовать, как готовые конструкции. Структура Uses-фразы: **Uses** *имя-модуля-1, имя-модуля-2, ..., имя-модуля-n***;**

Разделы объявлений программы (описательная часть)

Объявление меток

Меткой называется любое целое число без знака, либо обычный идентификатор, с помощью которого в исполнительной части программы можно пометить отдельные операторы для быстрого перехода к ним. Оператор отделяется от метки двоеточием (**:**).

Label *имя-метки-1, имя-метки-2, ..., имя-метки-n***;**

Объявление констант

Под *константой* понимается конструкция языка, значение которой в исполнительной части программы меняться не может. Структура данного раздела объявлений:

Const *имя-константы-1 = значение***;**
*имя-константы-2 = значение***;**
……
*имя-константы-n = значение***;**

- Объявление типов данных

- Под *типом данных* понимается множество допустимых значений данных, а также совокупность операций над ними. Все типы данных языка TurboPascal можно разделить на стандартные (предопределенные) и пользовательские определяемые программистом. Описание пользовательских типов данных осуществляется в этом разделе объявлений.

Type *имя-типа-1 = описатель-типа***;**
*имя-типа-2 = описатель-типа***;**
……
*имя-типа-n = описатель-типа***;**

Объявление типизированных констант

Осуществляется также как и обычных констант, но с указанием их типа.

Const *имя-константы-1***:** *тип-константы = значение***;**
*имя-константы-2***:** *тип-константы = значение***;**
……
*имя-константы-n***:** *тип-константы = значение***;**

- Объявление переменных

- В отличие от констант значения переменных в исполнительной части программы могут изменяться. Для их описания используют следующий раздел объявлений.

Var *имя-переменной-1A*, *имя-переменной-2A*: *тип-A*;
имя-переменной-1B, *имя-переменной-2B*: *тип-B*;…

- Объявление процедур

Процедурой называется подпрограмма, предназначенная для выполнения какой-либо законченной последовательности действий. Структура процедуры такая же, что и у основной программы (заголовок, разделы объявлений процедуры, исполнительная часть процедуры).

Procedure *имя-процедуры*(*список формальных параметров*);
<описания процедуры>
begin
<операторы тела процедуры>
end;

Список формальных параметров может содержать параметры различных категорий: параметры-значения, параметры-переменные, параметры-константы, параметры без типа, параметры-процедуры и параметры-функции.

- Объявление функций

Функцией называется подпрограмма, предназначенная для выполнения какой-либо законченной последовательности действий и возврата в вызывающую программу значения какого-либо параметра. Структура функции такая же, что и у основной программы (заголовок, разделы объявлений, операторы тела функции).

Function *имя-функции* (*список формальных параметров*): *тип-функции*;
<описания функции>
begin
<операторы тела функции>
end;

Список формальных параметров у функции формируется по тем же правилам, что и у процедуры.

Типы данных

Простые типы данных

1 *Целые типы* (предопределенные)

Идентификатор	Длина	Диапазон принимаемых значений
Integer	2 байта	-32768…+32767
Shortint	1 байт	-128…+127
Longint	4 байта	-2147483648…+2147483647
Byte	1 байт	0…255
Word	2 байта	0…65535

2 *Символьный тип* – стандартный тип языка **Char** определяется значениями полного набора ASCII-символов и занимает в памяти 1 байт.

3 *Логический тип* **Boolean** (1 байт) представляет собой тип данных, любой элемент которого может принимать лишь одно из двух возможных значений: *True* и *False*.

4 *Перечисляемый тип* – определяется пользователем с помощью набора идентификаторов, с которым могут совпадать значения параметра, объявленного данным типом.

Type *имя-типа* = (*значение1, значение2, ... значение-n*);

5 *Тип-диапазон* – задается пользователем с помощью указания наименьшего и наибольшего значений, разделенных двумя точками (**..**), которые могут принимать параметры данного типа.

Type *имя-типа* = *минимальное-значение* **..** *максимальное-значение***;**

6 *Вещественные типы*

Идентификатор	Длина	Число значащих цифр	Диапазон принимаемых значений
Real	6 байт	11–12	$\pm(2{,}9\times10^{-39}\ldots1{,}7\times10^{38})$
Single	4 байта	7–8	$\pm(1{,}5\times10^{-45}\ldots3{,}4\times10^{38})$
Double	8 байт	15–16	$\pm(5{,}0\times10^{-324}\ldots1{,}7\times10^{308})$
Extended	10 байт	19–20	$\pm(1{,}9\times10^{-4951}\ldots1{,}1\times10^{4932})$
Comp	8 байт	19–20	$-2^{63}+1\ldots2^{63}-1$

Все простые типы данных за исключением, вещественных, называются *порядковыми*.

Структурированные типы

1) *Массивы* – тип данных, представляющий собой совокупность фиксированного числа однотипных компонент. Каждая компонента массива имеет свой индекс, поэтому их иногда называют индексированными переменными. Объявляют тип данных массив следующим образом:

Type *тип-массив* = **array[***тип-индекса-1,тип-индекса-2,..., тип-индекса-n***] of** *тип-элементов*;

2) *Множество* – задает интервал значений, который является множеством всех подмножеств базового типа (базовым типом может быть любой порядковый тип, состоящий не более чем из 256 элементов) и определяется так:

Type *тип-множество* **= set of** *базовый-тип*;

Над множествами выполняют те же операции, что и над математическими множествами: проверки принадлежности, объединения, разности, пересечения.

3) *Строковый тип* данных *String* определяет последовательность ASCII-символов произвольной длины (от 0 до 255 символов), задаваемой при описании типа.

Type *тип-строка* = **string[***максимально возможная длина сроки*].

Над строками выполняют операции конкатенации или сцепления – (*строка*: = *строка-1* + *строка-2* + ... + *строка-n*;**)**, сравнения и обрабатывают с помощью стандартных функций языка.

4) *Запись* представляет собой такой тип данных, который содержит определенное число элементов (полей) и является смесью разных типов.

Type *тип-запись* **= record**
*переменная-1***:** *тип-1***;**
……
*переменная***:** *тип-n*
end;

5) Файловый тип представлен в языке TurboPascal в трех вариантах: текстовые файлы, типизированные файлы, нетипизированные файлы:

текстовый файл – **Var** *переменная***: text;**
типизированный файл – **Var** *переменная***: file of** тип компонента;
нетипизированный файл – **Var** *переменная***: file;**

Текстовый файл состоит из строк символов и является файлом формата DOS. Типизированный файл записывается в машинных кодах и имеет жесткую структуру, как правило, типа Record. Нетипизированные файлы используются в основном в тех случаях, когда файловая переменная необходима для операций ввода-вывода.

Указатель

Указатель дает адрес объекта определенного типа, называемого базовым типом. При определении типа-указателя используется этот базовый тип, перед которым ставится признак указателя – знак ^.

Type *тип-указатель* = **^***тип-переменной***;**

Var *переменная-указатель*: *тип-указатель***;**

Для работы с указателями используется адресный оператор @ и ряд стандартных конструкций языка.

Процедурный тип

Процедурный тип указывает, какой вид подпрограммы (процедуру или функцию) и с какими формальными параметрами можно использовать в качестве нового объекта программы (например, в виде формального параметра другой подпрограммы). Объявление процедурного типа похоже на заголовок подпрограммы.

Type *имя-типа* = **procedure(***список формальных параметров***);**

Type *имя-типа* = **function(***список формальных параметров***):** *тип-функции***;**

Далее можно вводить переменные этих типов.

Операции языка TurboPascal

- *Арифметические операции*: сложение (+), вычитание (–), умножение (*), деление (/), целочисленное деление (div), остаток деления (mod).

Логические операции

Not		**And**			**Or**			**Xor**		
A	**not A**	**A**	**B**	**A and B**	**A**	**B**	**A or B**	**A**	**B**	**A xor B**
False	True	False	False	False	False	False	False	False	False	False
True	False	False	True	False	False	True	True	False	True	True
–	–	True	False	False	True	False	True	True	False	True
–	–	True	True	True	True	True	True	True	True	False

Операции отношения: равно (=), не равно (<>), меньше (<), больше (>), меньше или равно (<=), больше или равно (>=).

Операторы языка TurboPascal

Все операторы языка можно разделить на две группы: простые и структурированные. К простым операторам относятся те, которые не содержат в себе других операторов. К ним относятся: оператор присваивания, пустой оператор, оператор вызова процедуры, оператор безусловного перехода GOTO. Структурированными называются операторы, которые могут включать в себя другие операторы. К этим операторам относятся следующие: составной оператор, условные оператор IF, оператор множественного выбора CASE, оператор цикла FOR, оператор цикла WHILE, оператор цикла REPEAT, оператор над записями WITH.

Оператор присваивания – присваивает переменной значение какого-либо выражения, при этом тип переменной должен быть совместим с типом выражения.

имя-переменной := *выражение*;

Пустой оператор – не выполняет никакого действия может быть отображен в программе в виде метки или точки с запятой, отделяющих пустой оператор от предыдущих или последующих операторов.

Оператор вызова процедуры – представляет собой оператор состоящий из имени вызываемой процедуры со списком фактических параметров (если они есть).

имя-процедуры **(***список фактических параметров***)**;

Оператор безусловного перехода GOTO – изменяет стандартную последовательность выполнения операторов и осуществляет переход к оператору, помеченному меткой.

Goto *метка*;

Составной оператор – представляет собой совокупность последовательно выполняемых операторов, заключенных в операторные скобки BEGIN и END.

Begin
Оператор-1;
Оператор-2;
……..
Оператор-N
End;

Условный оператор IF – является оператором разветвления и изменяет порядок выполнения операторов в зависимости от истинности или ложности некоторого логического условия (выражения). Этот оператор может быть записан в двух формах:

полной форме – **If** *условие* **Then** *Оператор-1* **Else** *Оператор-2*;

сокращенной форме – **If** *условие* **Then** *Оператор-1*;

Оператор-1 выполняется в случае истинности условия, а *Оператор-2* в случае ее ложности.

Оператор множественного выбора CASE – позволяет выбрать один вариант решения из некоторого количества вариантов.

Case *выражение* **of**
*Значение-1***:** *Оператор-1***;**
Значение-2 **..** *Значение-3***:** *Оператор-2***;**
*Значение-4***,** *Значение-5***,** *Значение-6***:** *Оператор-3***;**
……
Else *оператор-N*
End;

Выражение является выражением порядкового типа, а ветвь ELSE может отсутствовать. В данной структуре выполняется тот оператор, для которого фиксированное *Значение* совпадает со значением *Выражения.*

Оператор цикла FOR – осуществляет выполнение одного простого или составного оператора известное число раз. Существует два варианта записи этого оператора:

- инкрементная (с возрастанием параметра)
  For *Параметр-переменная* **:=** *Выражение-1* **To** *Выражение-2* **do** *Оператор***;**
- декрементная (с убыванием параметра)
  For *Параметр-переменная* **:=** *Выражение-1* **Downto** *Выражение-2* **do** *Оператор***;**

Переменная, являющаяся параметром цикла, может быть только порядкового типа и изменяется в каждой итерации на единицу. *Выражение-1* и *Выражение-2* определяют соответственно начальное и конечное значение параметра цикла.

Оператор цикла с предусловием WHILE –организует выполнение одного простого или составного оператора неизвестное заранее число раз до тех пор, пока сохраняется истинность логического выражения (условия завершения цикла).

While *Условие* **do** *Оператор***;**

Оператор цикла с постусловием REPEAT – организует выполнение цикла, состоящего из любого числа операторов, заранее неизвестное число раз до тех пор, пока логическое выражение (условие выхода из цикла), проверяемое в конце каждой итерации не примет значение True. В этом случае цикл завершает свою работу.

Repeat
*Оператор-1***;**
*Оператор-2***;**
……..
Оператор-N
Until *Условие***;**

Оператор над записями WITH – специальный оператор, позволяющий работать с объектами структурированного типа Record. При использовании этого оператора имя записи пишется только один раз, а внутри оператора указываются только имена полей записи.

With *Имя-записи* **do** *Оператор***;**

МОДУЛИ ЯЗЫКА TURBOPASCAL

В языке TurboPascal имеется большое количество стандартных подпрограмм, типов, констант и других конструкций, объединенных в несколько стандартных модулей. Они позволяют упростить процедуру написания программ и более полно использовать возможности компьютера и операционной системы. В частности ввод и вывод данных осуществляется при помощи стандартных процедур Read, ReadLn, Write, WriteLn. При составлении программ только стандартный модуль System автоматически подцепляется к программе, что позволяет пользоваться всеми его объектами, не делая для этого никаких дополнительных указаний в конструкции программы. Для использования всех остальных стандартных модулей (Crt, Graph, Dos, Strings, WinDos и др.) необходимо их присоединять к программе, используя для этого Uses-фразу.

Рассмотрим некоторые, наиболее часто используемые стандартные конструкции языка TurboPascal (для правильного использования подпрограмм необходимо ознакомиться со специальной справочной литературой, содержащей подробное описание заголовков подпрограмм).

Модуль SYSTEM

Этот модуль является основной библиотекой среды TurboPascal. В него входят все предопределенные процедуры и функции стандарта языка Паскаль, а также дополнительные подпрограммы, позволяющие выполнять разнообразные действия общего назначения.

- *Арифметические функции*:
- **Abs(***выражение***) –** абсолютное значение аргумента;
- **ArcTan(***выражение***)** – арктангенс аргумента;
- **Cos(***выражение***)** – косинус аргумента;
- **Exp(***выражение***)** – показательная функция с основанием *e* (экспонента);
- **Frac(***выражение***)** – дробная часть аргумента;
- **Int(***выражение***)** – целая часть аргумента;
- **Ln(***выражение***)** – натуральный логарифм аргумента;
- **Odd(***выражение***)** – проверяет аргумент на четность;
- **Pi** – число пи (3,1415926…);
- **Pred(***выражение***)** – предшествующее значение аргумента порядкового типа;
- **Random**[**(***число***)**] – возвращает случайное число вещественного или целого типа;
- **Sin(***выражение***)** – синус аргумента;
- **Sqr(***выражение***)** – квадрат аргумента;
- **Sqrt(***выражение***)** – квадратный корень аргумента;
- **Succ(***выражение***)** – следующее значение аргумента порядкового типа.
- *Арифметические процедуры:*
- **Dec(***V,N***)** – уменьшает переменную порядкового типа *V* на величину *N*;
- **Inc(***V, N***)** – увеличивает переменную порядкового типа *V* на величину *N.*
- *Процедуры и функции преобразования типов:*
- **Str(***N:p:q, S***) –** процедура, преобразующая число *N* в строку символов *S* (*p* и *q* – формат чисел, *p* – общее количество знаков числа, *q* – число знаков после запятой для вещественных чисел);
- **Val(***S, N, E***)** – процедура, преобразующая символьное представление числа *S* в числовую форму *N*, *E* – код ошибки;
- **Chr(***число***)** – функция, возвращающая символ с указанным ASCII-кодом;
- **Ord(***выражение***)** – функция, возвращающая порядковый номер значения выражения порядкового типа;
- **Round(***выражение***)** – функция, округляющая вещественное число до целого;
- **Trunc(***выражение***)** – функция, выделяющая целую часть выражения.
- *Процедуры и функции работы со строками:*
- **Delete(***S, N, I***)** – процедура, удаляющая из строки *S* ее часть, начиная с символ *N* длиною *I*;
- **Insert(***S1, S2, N***)** – процедура, вставляющая в строку *S2* подстроку *S1,* начиная с символа *N*;
- **Concat(***S1, S2,...,Sn***)** – функция, объединяющая строки *S1, S2,...,Sn* в одну;

- **Copy(***S, N, I***)** – функция, выделяющая из строки *S I* символов, начиная с *N*-ого.
- **Length(***S***)** – функция, определяющая текущую длину строки *S*;
- **Pos(***S1, S2***)** – функция, определяющая номер позиции, с которой начинается подстрока *S1* в строке *S2*.

Процедуры управления программой:

Break – процедура, осуществляющая немедленный выход из цикла;

Continue – процедура, осуществляющая досрочное начало новой итерации в цикле;

Exit – процедура, осуществляющая выход из текущей подпрограммы или программы;

Halt – процедура, прекращающая выполнение программы.

- *Процедуры ввода-вывода:*
- **Read(***список-ввода***)** – процедура, осуществляющая ввод исходных данных в соответствии со списком;
- **ReadLn(***список-ввода***)** – процедура, осуществляющая ввод исходных данных в соответствии со списком, переводя после ввода курсор на новую строку;
- **Write(***список-вывода***)** – процедура, осуществляющая вывод результатов в соответствии со списком;
- **WriteLn(***список-вывода***)** – процедура, осуществляющая вывод результатов в соответствии со списком, переводя после вывода курсор на новую строку.
- *Процедуры управления динамической памятью:*
- **Dispose(***переменная-указатель***)** – освобождает память, занятую динамической переменной;
- **FreeMem(***переменная-указатель, размер***)** – удаляет динамическую переменную заданного размера;
- **GetMem(***переменная-указатель, размер***)** – создает новую динамическую переменную заданного размера;
- **New(***переменная-указатель***)** – создает новую динамическую переменную.
- *Процедуры для управления вводом-выводом:*
- **Append(***файловая-переменная***)** – открывает существующий текстовый файл для добавления в конец информации;
- **Assign(***файловая-переменная, имя-файла***)** – связывает файловую переменную с внешним файлом;

BlockRead(*файловая-переменная, буфер, число-компонент, результат***)** – читает компоненты из файла без типа;

BlockWrite(*файловая-переменная, буфер, число-компонент, результат***)** – записывает компоненты в файл без типа;

- **Close(***файловая-переменная***)** – закрывает открытый файл;
- **Read(***файловая-переменная, список-ввода***)** – читает информацию из типизированного файла;
- **ReadLn(***файловая-переменная, список-ввода***)** – читает информацию из типизированного файла и переходит к следующей строке файла;
- **Reset(***файловая-переменная***)** – открывает существующий файл;
- **Rewrite(***файловая-переменная***)** – открывает новый файл;
- **Seek(***файловая-переменная***,** *номер-позиции***)** – настраивает файл на требуемую компоненту;
- **Write(***файловая-переменная, список-вывода***)** – записывает информацию в файл;
- **WriteLn(***файловая-переменная, список-вывода***)** – записывает информацию в файл и выводит в него признак конца строки.
- *Функции для управления вводом-выводом:*
- **Eof(***файловая-переменная***)** – фиксирует конец файла;
- **Eoln(***файловая-переменная***)** – фиксирует конец строки текстового файла;
- **FilePos(***файловая-переменная***) –** возвращает номер текущей компоненты файла;
- **FileSize(***файловая-переменная***)** – возвращает текущий размер файла в компонентах;
- **IOResult** – возвращает код выполнения операции ввода-вывода.

Модуль CRT

Этот модуль содержит константы, переменные и подпрограммы, предназначенные для работы с консолью в текстовом режиме. Рассмотрим основные подпрограммы модуля Crt.

Процедуры:
ClrScr – очищает текущее окно на экране;
Delay(*время***)** – задает задержку выполнения программы в миллисекундах;
GotoXY(*x, y***)** – перемещает курсор к элементу экрана с заданными координатами;
TextBackground(*цвет***)** – задает цвет фона;
TextColor(*цвет***)** – задает цвет символов;
Window(*x1, y1, x2, y2***)** – задает размеры окна на экране.
Функции:
KeyPressed – анализирует нажатие клавиш на клавиатуре;
ReadKey – считывает символ с клавиатуры;
WhereX – возвращает текущую координату *X* курсора;
WhereY – возвращает текущую координату *Y* курсора.

Модуль GRAPH

Этот модуль содержит типы, константы, переменные и подпрограммы, предназначенные для работы с консолью в графическом режиме. Количество предопределенных объектов в этом модуле велико. Одних процедур и функций в этом модуле более 80. Вот лишь некоторые основные процедуры:
Arc(*x, y, угол*1*, угол*2*, радиус***)** – построение дуги окружности;
Bar(*x*1*, y*1*, x*2*, y*2**)** – построение прямоугольника;
Bar3D(*x*1*, y*1*, x*2*, y*2*, ширина, грань***)** – построение параллелепипеда;
Circle(*x, y, радиус***)** – построение окружности;
ClearDevice – очистка графического экрана;
CloseGraph – завершение работы в графическом режиме;
FloodFill(*x, y, цвет-границы***)** – заполнение замкнутой области текущим орнаментом;
InitGraph(*драйвер, режим, путь***)** – инициализирует графический режим работы;
Line(*x1 ,y1, x2, y2***)** – построение линии от точки до точки;
OutText(*строка***)** – вывод на экран строки символов;
PutPixel(*x, y, цвет***)** – построение точки;
SetBkColor(*цвет***)** – задание цвета фона экрана;
SetColor(*цвет***)** – задание текущего цвета;
SetFillStyle(*параметр-заполнения, цвет***)** – задание параметров заполнения фигур.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1 Острейковский В. А. Информатика: Учебник для вузов. М.: Высш. шк.,1999. 511с.
2 Информатика. Базовый курс: Учебник для вузов / Под ред. С. В. Симонович СПб: Питер, 1999. 640 с.
3 Каймин В. А. Информатика: Учебник для вузов. М.: ИНФРА-М, 2000. 232 с.
4 Могилев А. В. и др. Информатика: Учеб.пособие для пед. учеб. заведений / Под ред. Е. К. Хеннера М.: Академия, 2000. 816 с.
5 Информатика: Энциклопед. словарь для начинающих / Под общ. ред. Д. А. Поспелова М.: Педагогика Пресс, 1994. 352 с.
6 Экономическая информатика / Под ред. П. В. Конюховского, Д. Н. Колесова СПб: Питер, 2000. 560 с.
7 Информатика для юристов и экономистов / Под ред. С. В. Симоновича СПб: Питер, 2001. 688 с.
8 Информатика: Учебник для вузов / Под ред. Н. В. Макаровой 3-е изд., перераб. М.: Финансы и статистика, 2001. 768 с.
9 Фигурнов В. Э. IBM PC для пользователя. 7-е изд., перераб. и доп. М.: Инфра-М, 1997. 640 с.
10 Назаров С. В., Першиков В. И., Тафинцев В. А. и др. Компьютерные технологии обработки информации: Учеб. пособие. М.: Финансы и статистика, 1995. 248 с.
11 Справочное руководство для пользователей компьютеров IBM PC: MS DOS 6.0. М.: ВА Принт, 1994. 318 с.
12 Аглицкий Д. С., Любченко С. А. Персональный компьютер и WINDOWS 95 для всех. М.:"ДИС", 1997. 368 с.
13 Компьютерные сети: Учебный курс: Пер. с англ. 2-е изд., испр. и доп. М.: Изд. отдел "Рус. редакция" ТОО "Channel Traing Ltd", 1998. 696 с.

14 Левин А. А. Самоучитель работы на компьютере. 5-е изд., испр. и доп. М.: Нолидж, 1999. 542 с.
15 Платонов Ю. М., Уткин Ю. Г. Диагностика и ремонт персональных компьютеров. М.: Радио и связь, 1996. 208 с.
16 Жаров А. В. "Железо" IBM 99 или все о современном компьютере. 6-е изд., испр. и доп. М.: МикроАрт, 1999. 352 с.
17 Беньяш Ю. Л. Освоение персонального компьютера и работа с документами. М.: Радио и связь, 1999. 408 с.
18 Анин Б. Ю. Защита компьютерной информации. СПб.: БХВ-Санкт-Петербург, 2000. 384 с.
19 Касперский Е. В. Компьютерные вирусы: что это такое и как с ними бороться. М.: СК Пресс, 1998. 288 с.
20 Синклер А. Большой толковый словарь компьютерных терминов: Русско-английский, англо-русский. М.: Вече, 1999. 512 с.
21 Олифер В. Г., Олифер Н. А. Компьютерные сети: Принципы, технологии, протоколы: Учеб. пособие для вузов. СПб.: Питер, 2001. 672 с.
22 Дьяконов В. И. Компьютер для студента: Самоучитель. СПб: Питер, 2000. 592 с.
23 Форсайт Р. Паскаль для всех. М.: Машиностроение, 1986. 288 с.
24 Справочник по процедурам и функциям Borland Pascal with Objects 7.0. Киев: Диалектика, 1993. 272 с.
25 Епанешников А. М., Епанешников В. А. Программирование в среде TurboPascal 7.0. М.: Диалог-МИФИ, 1995. 288 с.
26 Немнюгин С. А. TurboPascal: Практикум. СПб: Питер, 2001. 256 с.

Содержание