

**Современный
Гуманитарный
Университет**

Дистанционное образование

Рабочий учебник

Фамилия, имя, отчество _____

Факультет _____

Номер контракта _____

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

ЮНИТА 2

**СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
ПРОЦЕССОВ РАЗРАБОТКИ И СОПРОВОЖДЕНИЯ
ПРОГРАММНЫХ КОМПЛЕКСОВ**

МОСКВА 2000

Разработано А.В. Карпейкиным, к.т.н., с.н.с.

Рекомендовано Министерством
общего и профессионального
образования Российской Федерации
в качестве учебного пособия для
студентов высших учебных заведений

КУРС: СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Юнита 1. Системное программное обеспечение.

Юнита 2. Системное программное обеспечение процессов разработки и сопровождения программных комплексов.

Юнита 3. Системное программное обеспечение сетей ЭВМ.

ЮНИТА 2

Рассмотрены средства разработки программного обеспечения, процессы компиляции, интерпретации и отладки программных средств (ПС), макрокоманды, дано описание интегрированных сред разработки ПС. Дано понятие CASE-систем и приведены примеры их использования, а также изложены понятия баз данных (БД) и систем управления БД (СУБД). Рассмотрены информационные сети, процессы передачи данных и взаимодействия открытых систем, а также вопросы защиты информации от вирусов и несанкционированного доступа.

Для студентов Современного Гуманитарного Университета

Юнита соответствует профессиональной образовательной программе № 1

ОГЛАВЛЕНИЕ

ДИДАКТИЧЕСКИЙ ПЛАН	4
ЛИТЕРАТУРА	5
ПЕРЕЧЕНЬ УМЕНИЙ	6
ТЕМАТИЧЕСКИЙ ОБЗОР	8
1. Системное ПО разработки программных средств	8
1.1. Интерпретаторы	8
1.2. Компиляторы	9
1.3. Интегрированная среда разработки	11
1.4. Отладка программных средств	18
1.5. Макрогенераторы	22
1.6. Системы управления базами данных	28
1.6.1. Автоматизированные банки данных	28
1.6.2. Объекты и отношения объектов	31
1.6.3. Данные	32
1.6.4. Модель и подмодель данных	34
1.6.5. Функционирование СУБД	37
1.7. CASE-системы	43
1.7.1. Понятие CASE-систем	43
1.7.2. Модели жизненного цикла ПО	46
1.7.3. CASE-средства. Общая характеристика и классификация	49
1.7.4. Оценка и выбор CASE-средств	51
2. Системное ПО передачи и защиты данных	54
2.1. Передача данных	57
2.1.1. Коммутация каналов	58
2.1.2. Коммутация сообщений	60
2.1.3. Коммутация пакетов	61
2.1.4. Сопряжение ЭВМ и устройств в сетях	62
2.2. Информационные сети	62
2.2.1. Классификация ЛВС	63
2.3. Взаимосвязь открытых систем	66
2.3.1. Уровни взаимодействия в ЛВС	67
2.4. Вирусы и антивирусное ПО	70
2.4.1. Общие сведения о вирусах	70
2.4.2. Механизм действия вирусов	74
2.4.3. Антивирусное ПО	80
2.5. Защита от несанкционированного доступа	84
2.6. Технологическая безопасность	87
ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ	95
ТРЕНИНГ УМЕНИЙ	96
ГЛОССАРИЙ*	

* Глоссарий расположен в середине учебного пособия и предназначен для самостоятельного заучивания новых понятий.

ДИДАКТИЧЕСКИЙ ПЛАН

Системное ПО разработки программных средств.

Интерпретаторы. Компиляторы. Интегрированная среда разработки. Отладка ПС. Макрогенераторы. Системы управления базами данных. CASE-системы.

Системное ПО передачи и защиты данных.

Передача данных. Информационные сети. Взаимосвязь открытых систем. Вирусы и антивирусное ПО. Защита от несанкционированного доступа. Технологическая безопасность.

ЛИТЕРАТУРА

Базовая

- *1. Острейковский В.А. Информатика. М.: Высшая школа, 1999.
- *2. Информатика: Учебник / Под. ред. Н.В. Макаровой. М.: Финансы и статистика, 1999.

Дополнительная

- *3. Блэк Ю. Сети ЭВМ: протоколы, стандарты, интерфейсы. М.: Мир, 1990.
- *4. Верлань А.Ф. Широчкин В.П. Информатика и ЭВМ. Киев: Техника, 1987.
- *5. Калянов Г.Н. CASE. Структурный системный анализ (автоматизация и применение). М.: Лори, 1996.
- *6. Международные стандарты, поддерживающие жизненный цикл программных средств. М.: МП "Экономика", 1996.
- 7. Задков В.Н. Пономарев Ю.В. Компьютер в эксперименте, архитектура и программные средства систем автоматизации. М.: Наука, 1988.
- 8. Мячев А.А. Персональные ЭВМ: Краткий энциклопедический справочник. М.: Финансы и статистика, 1992.
- 9. Брябрин В.М. Программное обеспечение персональных ЭВМ. М.: Наука, 1989.

Примечание. Знаком (*) отмечены работы, на основе которых составлен тематический обзор.

Современный Гуманитарный Университет

ПЕРЕЧЕНЬ УМЕНИЙ

№ п/п	Умение	Алгоритмы
1.	Добавление (удаление) модуля в проект (из проекта) в Delphi	<ol style="list-style-type: none"> 1. Открыть проект. 2. В меню «View» выбрать пункт «Project Manager». 3. Добавление или удаление модуля: <ol style="list-style-type: none"> 3.1. Для добавления модуля в окне «Project Manager» нажать кнопку «Add» и перейти к пункту 4. 3.2. Для удаления модуля из проекта сначала надо выбрать удаляемый модуль, а затем в окне «Project Manager» нажать кнопку «Remove» и перейти к пункту 6. 4. В появившемся диалоговом окне ввести имя нового модуля или выбрать существующий модуль. 5. Нажать кнопку «ОК». 6. Завершить работу с окном «Project Manager».
2.	Написание ассемблерной макрокоманды	<ol style="list-style-type: none"> 1. Создать заголовок макрокоманды и при необходимости определить формальные параметры. 2. Описать тело макрокоманды. 3. Определить конец макрокоманды. 4. Определить в программе макровывоз.
3.	Выбор интерфейса сопряжения ЭВМ и периферийных устройств	<ol style="list-style-type: none"> 1. Определить тип вычислительной системы – распределенная или многомашинная. 2. Определить, на каких расстояниях будут находиться ЭВМ. 3. Определить, какие периферийные устройства будут использоваться. 4. Выбрать вид интерфейса.

№ п/п	Умение	Алгоритмы
4.	Выбор канала связи для информационной сети	<ol style="list-style-type: none"> Исходя из объема и времени загрузки канала выбрать режим работы: <ol style="list-style-type: none"> Коммутируемый – перейти к пункту 2. Некоммутируемый – завершить выбор. Определить режим передачи: <ol style="list-style-type: none"> Непрерывный – выбирается канал связи с коммутацией канала. Прерывистый – перейти к пункту 3. Определить требуемую быстроту передачи сообщения. <ol style="list-style-type: none"> С минимальной затратой времени – выбирается канал с коммутацией пакетов. Затраты времени не столь важны – выбирается канал с коммутацией сообщений.
5.	Выбор физических носителей сигналов в ЛВС	<ol style="list-style-type: none"> Определить требуемую скорость передачи данных: <ol style="list-style-type: none"> Максимально высокая – выбирается оптоволоконный кабель. Высокая – в зависимости от стоимости выбирается многожильный или коаксиальный кабель (соответственно, высокая и невысокая стоимость). Невысокая – выбирается витая пара. Определить степень защиты информации в кабеле: <ol style="list-style-type: none"> Высокая – выбирается оптоволоконный кабель. Средняя – выбирается многожильный или коаксиальный кабель. Низкая – выбирается витая пара. На основе пунктов 1 и 2 делаем окончательный выбор.

ТЕМАТИЧЕСКИЙ ОБЗОР*

1. СИСТЕМНОЕ ПО РАЗРАБОТКИ ПРОГРАММНЫХ СРЕДСТВ

1.1. Интерпретаторы

Еще в конце 70-х – начале 80-х ЭВМ были не столь мощными, как сейчас, обладали очень малым объемом оперативной памяти (до 64 кБ, что объяснялось ее большой стоимостью). Это вызывало большие сложности для выполнения задач программирования. В столь малом объеме оперативной памяти размещалась операционная система и загружалась программа-интерпретатор. Для того, чтобы не перегружать оперативную память, исходный текст программы не транслировался в машинный код, а язык программирования состоял как бы из двух частей – редактора (в котором набирался и модифицировался исходный текст программы) и интерпретатора, исполняющего подготовленные программы. **Интерпретатор** – программа или система, анализирующая строку из языка программирования и выполняющая её, т.е. выполняющая интерпретацию. Особенностью интерпретирующей системы является пооператорный анализ и перевод на машинный язык каждого из операторов выполняемой программы без сохранения в памяти машинных кодов (объектного модуля). Язык программирования, допускающий выполнение программы в режиме интерпретации высказываний данного языка, называют **интерпретируемым языком**. К интерпретируемым языкам программирования относятся, прежде всего, ранние версии ассемблеров и Бейсика.

Интерпретация используется в логическом программировании и продукционных системах. Так, например, различают **интерпретацию «от фактов»** (процедурную интерпретацию правила вида «если А, то В», при которой добавление в базу знаний факта «А» вызывает добавление в нее факта «В») и **интерпретацию «от цели»** (процедурную интерпретацию правила вида «если А, то В», при которой для достижения цели «В» делается попытка достичь цели «А»).

Исполнение программы начинается по команде из оболочки языка программирования (среды программирования). С этого момента и до конца работы программы, например, при остановке по **ошибке** (событии, заключающемся в потере, повреждении или искажении информации; она может быть вызвана отказом или сбоем объекта, а также неправильными действиями пользователя, дефектами программы, носителей информации) или по требованию потребителя, пользователь уже не меняет текста программы, а только наблюдает за ходом интерпретации, в ходе которой может происходить обмен информацией (рабочий диалог) между человеком и ЭВМ – например, процедура ввода-вывода и др. (рис. 1.1).

* Жирным шрифтом выделены новые понятия, которые необходимо усвоить. Знание этих понятий будет проверяться при тестировании.

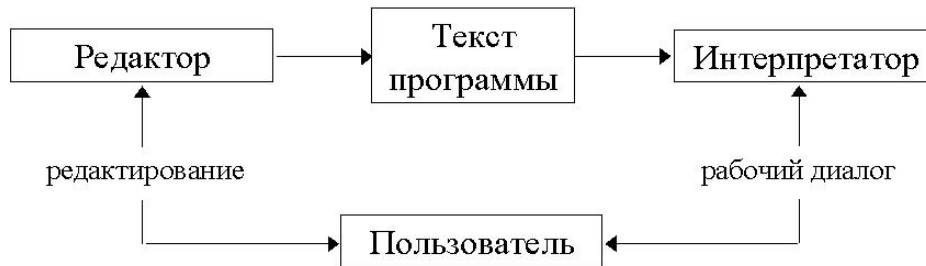


Рис. 1.1

Отметим, что в режиме интерпретации оперативная память дополнительно занимается лишь данными из выполняемой строки. Именно поэтому интерпретируемые языки в условиях малого объема оперативной памяти позволяют пользователю модифицировать и перезапускать программу без необходимости ее повторной трансляции целиком, а также останавливать программу в любой точке, анализировать или модифицировать любые ее переменные и возобновлять выполнение в этой же точке, т.е. осуществлять процесс отладки программы.

Но подобная гибкость имеет существенный недостаток: интерпретаторы и выполняемые с их помощью программы работают очень медленно (это особенно видно при выполнении циклов), т.к. каждая строка интерпретируется и выполняется по мере ее нахождения.

1.2. Компиляторы

Преодолеть недостатки интерпретаторов в части быстродействия позволяют компиляторы. **Компилятор** – это системная программа, осуществляющая перевод прикладной программы с языка высокого уровня (формального языка) на машинный без одновременного ее выполнения. В результате компиляции получается объектный модуль программы.

В состав систем программирования также входит **исполняющая система** – совокупность подпрограмм, при обращении к которым транслируются некоторые сервисные программы, к которым программа обращается во время работы, например, файловые операции или операции со строками.

Процесс компиляции обычно включает в себя следующие этапы:

- Считывание исходной программы и последовательный – оператор за оператором – ее анализ.
- Лексический анализ входного текста, в процессе которого исходная программа переводится на стандартный входной язык компилятора и преобразуется к виду, удобному для работы на этапах синтаксического и семантического анализов. Это

закljučается в удалении из исходного текста программы пробелов и комментариев, в сборке из отдельных символов строк идентификаторов, служебных слов и т.д.

- Построение таблиц символов, операторов и внешних ссылок, необходимых для последующего анализа:
 - с помощью **синтаксического анализатора** осуществляется анализ структуры каждого оператора и проверяется наличие синтаксических ошибок в них;
 - анализ затребуемых в программе структур данных и подготовка сведений, необходимых для распределения памяти под переменные, массивы и т.д.;
 - анализ логической схемы исходной программы и глобальный анализ ошибок, например, проверка операторов завершения или выхода из цикла.
- Исследование каждого предложения входного языка и генерация в промежуточной символической внутренней форме предложений, семантически эквивалентных предложениям исходной программы.
- Производится компоновка и оптимизация программы (в том числе и информации из модулей или компилируемых модулей, переменные, процедуры и функции которых используются в программе). Под **компилируемым модулем** понимается описание или текст модуля программы, предназначенное для компиляции в качестве самостоятельного текста, и имеющего в своем исходном тексте процедуры функции и подпрограммы, использующиеся в теле основной программы.
- Генерация с промежуточного языка объектного кода модуля и/или программы, готовой к исполнению (**исполняемого файла**).

Отметим, что в исполняемом файле есть **точки выхода** – команды в программе, завершающие ее выполнение, а также **точки рестарта** – места в программе, с которых при необходимости может быть возобновлено повторное выполнение программы или ее части.

Время, в течение которого программа, записанная на языке высокого уровня, транслируется в машинный код, называют **временем компиляции**.

Особо стоит остановиться на таком этапе компиляции, как компоновка и оптимизация. Хорошим стилем программирования уже давно стало использование модульного принципа построения программы, т.е. модулей. Использование уже отлаженных модулей, выполняющих определенные функции и процедуры (например, операции с матрицами, комплексными числами и т.п.), позволяет значительно сократить время написания программы. Кроме того, каждая программа использует некоторые системные функции операционной системы, а также стандартные функции и процедуры из библиотеки языка программирования. Компоновка и оптимизация применительно к модулям заключается: а) во встраивании их тел (или частей тел, если

из модуля используются не все процедуры и функции) в исходный текст исполняемой программы; б) расстановке адресов **точек вызова** (место в исполняемой программе, в которой находится ссылка на процедуры и функции модулей или подпрограмм) и **точек повторного входа** (адресов или меток операторов, с которых начинается повторное выполнение процедуры, функции или подпрограммы).

Основными недостатками компиляторов в конце 70-х – начале 80-х годов (в связи с малым объемом оперативной памяти у ЭВМ) являлись:

1. Большой объем откомпилированной программы, поэтому компиляторы не взаимодействовали непосредственно с оболочкой языка программирования. То есть сначала набирался исходный текст программы, а затем осуществлялась ее компиляция.
2. При обнаружении в процессе компиляции ошибок в исходной программе необходимо было войти в оболочку языка программирования (или текстовый редактор), исправить выявленные ошибки (в том числе и во время выполнения откомпилированной программы) и только затем осуществить новый процесс компиляции программы.

Поэтому в то время процесс отладки программы осуществлялся в режиме интерпретации (если язык программирования позволял это), а затем программа компилировалась и распространялась в виде исполнимого файла.

Возможности современных ЭВМ позволяют компиляторам, встроенным в среду языка программирования, исправлять ошибки в программе, выявленные при ее компиляции, а также осуществлять модификацию и отладку программы не выходя из среды (оболочки) языка программирования. Разработанные в последнее время интегрированные среды разработки (IDE – Integrated Development Environment) позволяют все это осуществить, а также предоставляют пользователю ряд дополнительных возможностей.

1.3. Интегрированная среда разработки

Под интегрированной средой разработки понимается среда языка программирования, включающая в себя в общем случае:

- многооконный редактор, обладающий такими функциями, как копирование, поиск, замена, вставка и др.;
- компилятор, позволяющий откомпилировать программу с различными опциями компиляции;
- отладчик программы, включающий в себя ряд сервисных средств;
- средства вызова сервисных программ, не входящих в среду языка программирования;
- средства для индивидуальной настройки среды языка программирования под конкретного пользователя;

- средства для вызова общей, оперативной и контекстной помощи (подсказки).

Первая интегрированная среда разработки была разработана фирмой Borland International в 1987 году для языка программирования Turbo Pascal 4.0 (Турбо Паскаль 4.0). В дальнейшем все основные языки (системы) программирования также включают в себя интегрированные среды. На рис. 1.2 изображен общий вид интегрированной среды разработки для среды Turbo Pascal 7.0, работающей в операционной системе MS-DOS. На рисунке видны: основное меню (вверху); строка статуса (внизу); диалоговое окно для установки опций компилятора (на переднем плане); окна редактора (на заднем плане). В окне редактора открыты два окна – одно с текстом исходной программы (текст в нем начинается со строки Program) и окно с исходным текстом подключаемого к программе модуля (текст в нем начинается со строки Unit).



Рис. 1.2

В настоящее время все основные языки программирования работают в среде многозадачных операционных систем (например, Windows-98, Windows NT) и имеют улучшенные интерфейсы (IDE). Поэтому более детальное рассмотрение интегрированной среды разработки проведем на примере IDE систем программирования фирмы Borland International Delphi 3.0 (Дельфи 3.0), т.к. IDE других языков программирования по своей структуре похожи и отличаются в незначительных мелочах.

На рис. 1.3 приведен общий вид IDE языка Дельфи 3.0. На рисунке видны: основное меню (вверху); панель быстрого доступа и палитра компонентов (под основным меню); инспектор объектов (Object

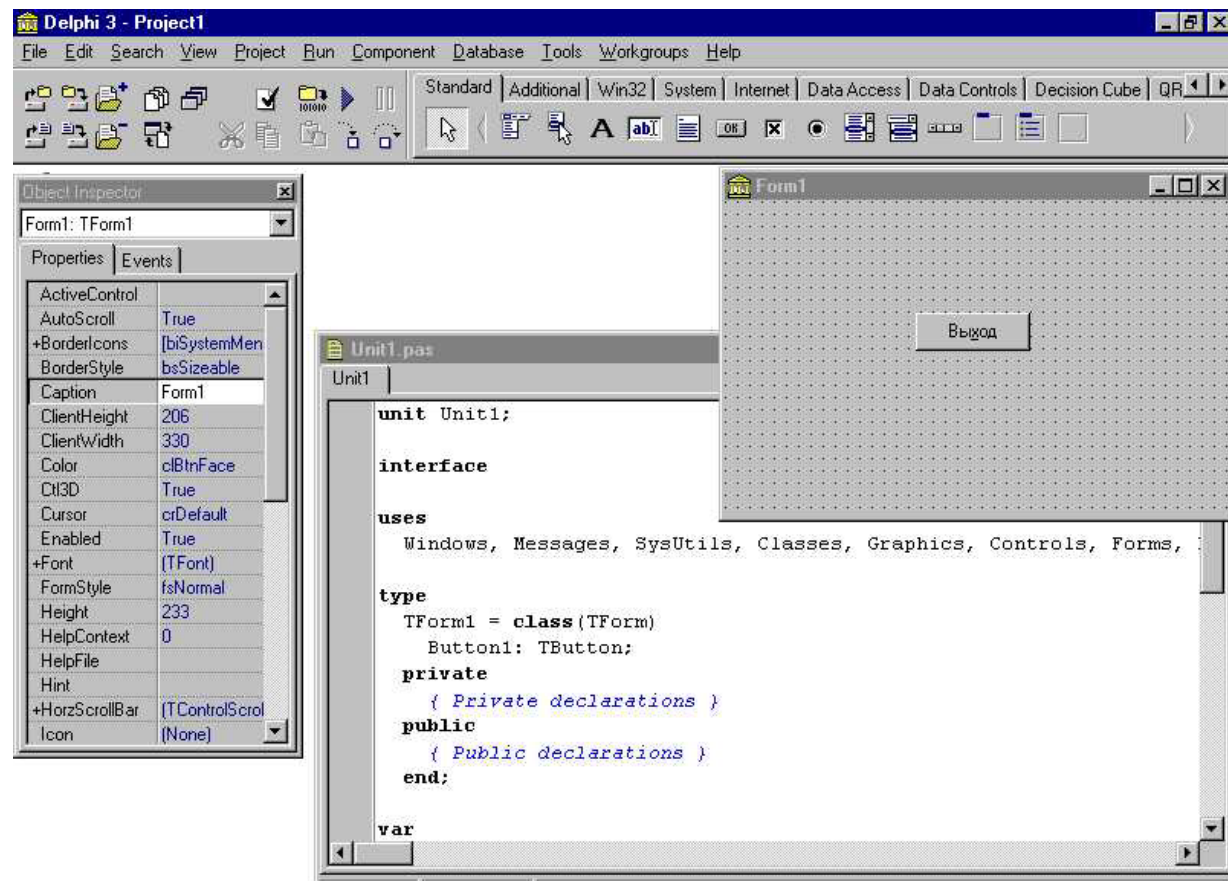


Рис. 1.3

Inspector) (в левой части рисунка); окно формы (в правой части рисунка – с заголовком Form1); окно редактора (в центре на заднем плане – с заголовком Unit1). Рассмотрим более детально каждый из компонентов интегрированной среды разработки Дельфи 3.0.



Рис. 1.4

Основное меню (рис. 1.4) содержит в себе основные команды среды IDE, редактора, компилятора и т.д., разбитые по разделам. В меню «File» (Файл) содержатся такие команды, как «Новое приложение», «Открыть», «Сохранить», «Новая форма», «Добавить в проект» (**проект** – это программа, рассматриваемая как целое в контекстах хранения, трансляции, объединения с другими программными модулями и загрузки в оперативную память для выполнения), «Удалить из проекта», «Печатать», «Выход» и некоторые другие. Меню «Edit» (Редактировать) включает в себя команды копирования, удаления, вставки, отмены последнего действия в редакторе или форме и т.п. Меню «Search» (поиск) содержит команды поиска, поиска во всех файлах проекта, замены и т.п. Меню «View» (Просмотр) содержит команды просмотра (с возможностью открытия) списка всех модулей, входящих в проект, в режиме отладки из этого меню можно открыть окна просмотра значений переменных, содержимого стека и т.п. Меню «Project» включает в себя команды компиляции проекта, проверки синтаксических ошибок (выполняется намного быстрее компиляции, т.к. при этом исполнимый файл программы не создается), информации об откомпилированном проекте, настройки опций проекта (на рис. 1.5 изображено диалоговое окно выставки опций проекта, открытое на закладке опций компилятора) и некоторых других.

Меню «Run» (Выполнить) содержит команды выполнения (запуска программы на выполнение без выхода из среды разработки), пошагового выполнения программы и добавления (редактирования) **точек прерывания** (место в программе, в котором она была прервана внешним сигналом или по условному событию в программе и с которой должно быть продолжено ее выполнение) в режиме отладки, добавления переменных в окно просмотра значений переменных и т.п. Меню «Component» (Компонент) позволяет добавить или удалить компоненты из библиотеки языка программирования. *Компонент языка программирования* – это визуальный или не визуальный объект (класс), обладающий определенными свойствами и методами: процедурами и функциями, обрабатывающими события от клавиатуры, мыши, исполняемой программы и операционной системы. Например, компонентами являются

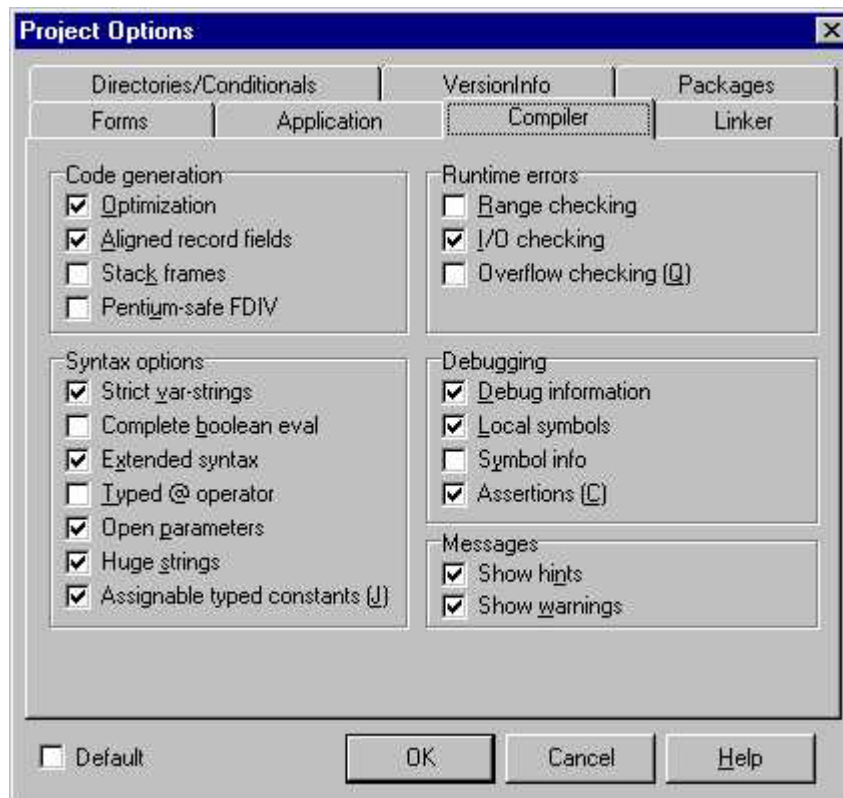


Рис. 1.5

рассматриваемое нами меню, кнопки, окна и др. Меню «Database» (База данных) включает в себя утилиты, необходимые при работе с программированием баз данных. Меню «Tools» (Инструменты) позволяет настроить внешний вид интегрированной среды разработки, редактора, а также вызвать исполнение утилит (добавить или удалить их из меню «Tools») и программ, не входящих в IDE. Меню «Workgroups» (Рабочие группы) позволяет синхронизировать версию редактируемой программы при работе над ней группы программистов (работе в сети). Меню «Help» (Помощь) содержит в себе подсказки по разным вопросам – по IDE, по разделам Дельфи и т.д.

Панель быстрого доступа (рис. 1.6) содержит в себе кнопки, дублирующие часть команд из основного меню, что значительно ускоряет работу. Отметим, что кнопки на панели быстрого доступа могут быть добавлены или удалены по желанию пользователя.



Рис. 1.6

Палитра компонентов (рис. 1.7) содержит в себе все компоненты, доступные данной версии языка программирования, а также дополнительно установленные пользователем.



Рис. 1.7

Окно формы (рис. 1.8) показывает разработанное в IDE визуальное представление модуля, т.е. то, что будет видно на экране во время выполнения программы (его описание видно в редакторе – модуль unit UnitCalc на рис. 1.9). Отметим, что параметры визуального представления – такие, как размер и заголовков формы, цвет и размер шрифта и др. (для различных визуальных компонентов свойства различаются) – хранятся в отдельном файле, а изменяются в инспекторе объекта (рис. 1.10)

Таким образом, современные интегрированные среды разработки позволяют значительно повысить эффективность и сократить время на разработку и отладку программного обеспечения.

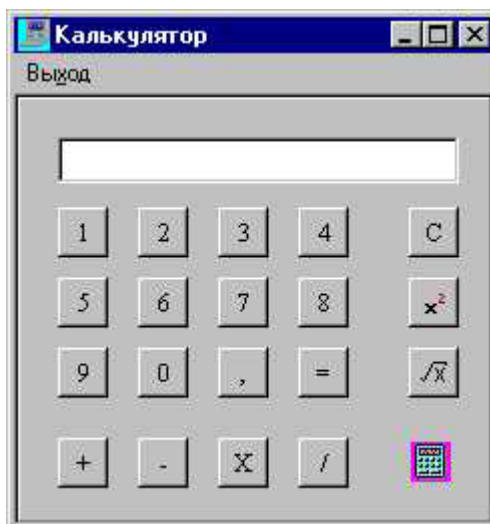


Рис. 1.8

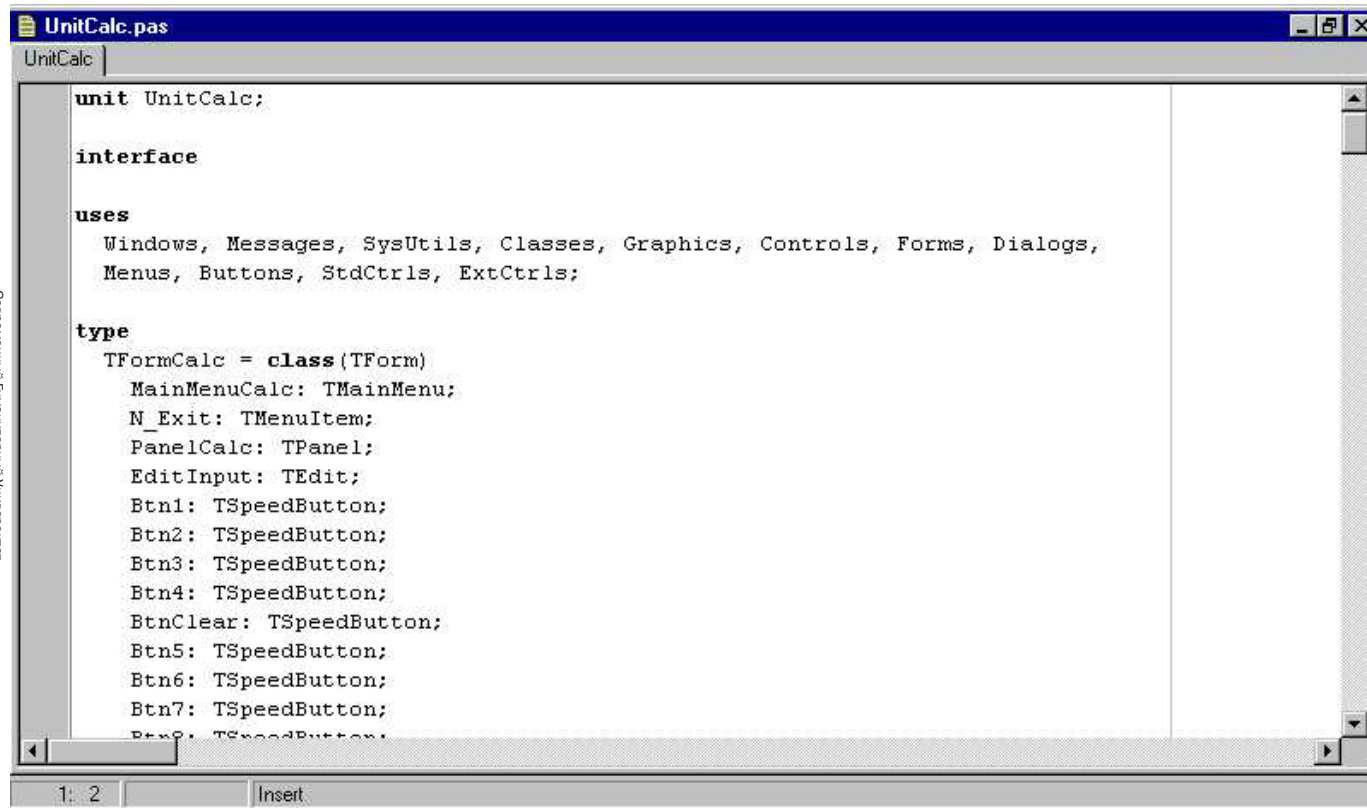


Рис. 1.9

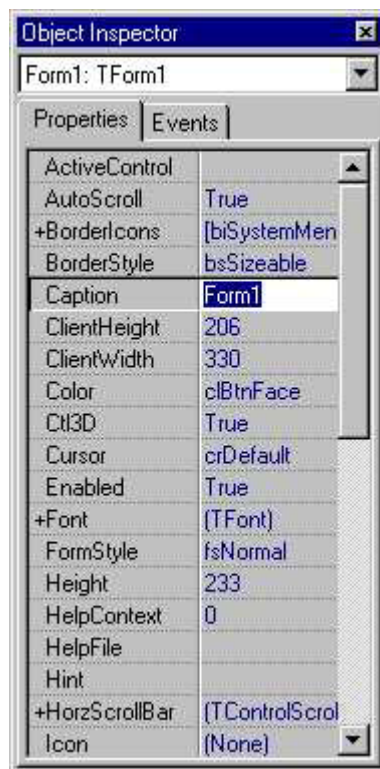


Рис. 1.10

1.4. Отладка программных средств

Вопрос о том, выполняет ли программа возложенные на нее функции, является основным. Если программа только что составлена, то в очень редких случаях она не содержит ошибок (речь идет, понятно, о достаточно объемных и сложных программах). Ошибки в программе необходимо найти и исправить. Они могут вызвать **ошибку программного изделия** (запись элемента программы или программной документации, использование которой приводит к неправильному результату; по степени влияния на результат обработки данных различают грубые, существенные и незначительные ошибки) или **отказ программного изделия** (события, заключающегося в проявлении неработоспособности программного изделия).

Вообще процесс написания и последующего сопровождения программы (жизненный цикл программного обеспечения – ЖЦ ПО) состоит из следующих этапов.

Анализ требований. Этот этап исключительно важен. В ходе этого этапа должны быть решены следующие вопросы:

Что должна делать программа?

В чем состоят реальные проблемы, разрешению которых она должна способствовать?

Что представляют собой входные данные?

Какими должны быть выходные данные?

Какими ресурсами располагает проектировщик?

Определение спецификаций. Требования к программе должны быть предъявлены в виде ряда спецификаций, явно определяющих рабочие характеристики разрабатываемой программы (например, скорость выполнения, гибкость применения и др.).

Проектирование. На этом этапе создается общая структура программы, которая должна удовлетворять спецификациям, определяются общие принципы управления и взаимодействия между различными компонентами программы. Или, другими словами, создается алгоритм функционирования программы.

Кодирование. Заключается в переводе алгоритма на язык программирования.

Тестирование. Заключается в проверке работоспособности программы; проверке правильности программы (т.е., что программа делает в точности то, для чего она предназначена); проверке ее эффективности и вычислительной сложности.

Сопровождение. Это этап эксплуатации системы. Каким бы ни было тестирование, но, к сожалению, в больших программных комплексах чрезвычайно тяжело устранить абсолютно все ошибки. Основная задача этого этапа – устранение ошибок, выявленных при эксплуатации программы. Но при этом решаются и другие задачи – выявление «узких» мест или неудачных (но при этом работающих правильно) проектных решений, а также обучение пользователей программы, их консультация, и снабжение новыми версиями программы.

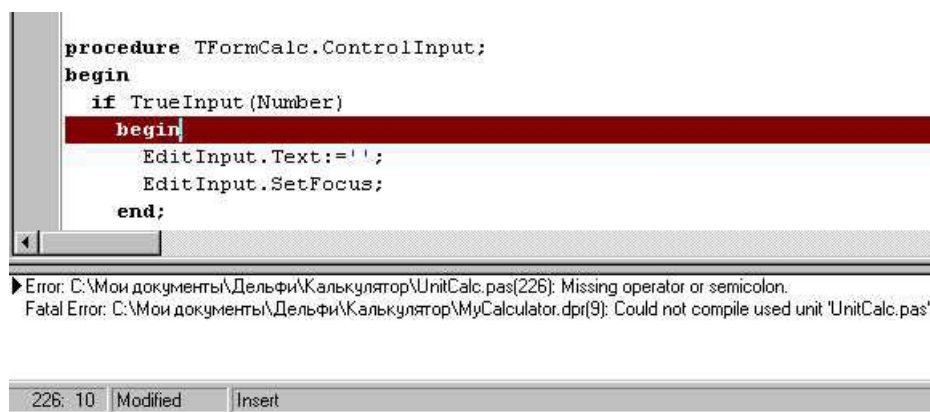
То есть, происходит процесс отладки (тестирования). Существуют программы для выявления ошибок и их локализации. *Отладка* – это процесс исправления программы таким образом, что при этом исправляются существующие ошибки и не вносятся новые. На практике процесс исправления ошибок без оказания при этом влияния на остальные части программы оказывается достаточно сложной задачей.

Можно выделить следующие виды ошибок:

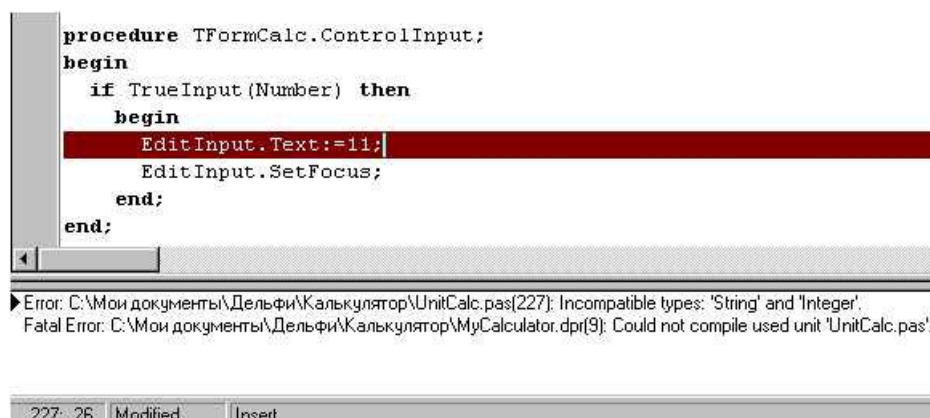
1. **Синтаксические**, т.е. несоответствие конструкций в программе правилам языка программирования.
2. **Программные**, т.е. это ошибки, возникающие при разработке программы. Они делятся на два типа – ошибки в алгоритме, возникающие при разработке технического задания или проектировании программы; и ошибки, возникающие при написании программы на языке программирования.

Синтаксические ошибки обнаруживаются на этапе компиляции

программы и обычно легко устраняются. Однако при этом полностью полагаться на компилятор нельзя – он лишь проверяет правильность конструкции и не во всех случаях, а ответственность за исправление несет программист. При синтаксической ошибке строка с ошибкой выделяется, а в строке статуса появляется номер ошибки и ее краткий комментарий (рис. 1.11 а, б). На рис. 1.11, а представлен случай, когда компилятор выделил не строку с ошибкой, а следующую – это объясняется тем, что конструкция *if <выражение> then* не содержит окончания *then*, поэтому компилятор ищет его на следующей строке и не находит. Рис. 1.11, б иллюстрирует тривиальный случай – несоответствие типов выражений в левой и правой частях выражения – типов строка и целое.



a)



б)

Рис. 1.11

Программные ошибки гораздо труднее обнаружить и локализовать. Ошибки в алгоритме не могут быть обнаружены ЭВМ, так как она не «знает» решаемой задачи. Это, например, так называемые за цикливания, когда программа не выходит из цикла и как бы за висает – в алгоритме не определено условие выхода из цикла. К ошибкам написания относится, например, присвоение в разных местах программы одной и той же переменной разных значений – это может привести, в частности, к операции деления на ноль.

Отладка программы начинается, как правило, с анализа правильности перевода алгоритма на язык программирования и правильности логики построения программы (эти действия выполняются, естественно, без запуска программы на выполнение). После исправления выявленных на этом этапе ошибок прибегают к **трассировке** – средству отладки программы, обеспечивающему индикации конкретного набора команд и программных объектов по мере обращения к ним в ходе выполнения программы (или, по-другому – к пошаговому выполнению программы).

Пошаговое выполнение может выполняться не только с самого начала программы, но и с точки прерывания (останова), что позволяет выполнить уже отлаженную или проверенную части программы до участка программы, где находится или начинает проявляться программная ошибка. При пошаговом выполнении можно увидеть и изменить значения переменных, увидеть процесс модификации переменных после выполнения с ними каких-либо действий и т.д. На рис. 1.12 показано окно просмотра значений переменных (слева, с заголовком Watch List) и окно редактора с кодом программы – в нем можно увидеть две выделенные линии: верхняя линия соответствует точке прерывания программы, а нижняя – текущему выполняемому оператору программы (шагу программы). На рис. 1.13 показано

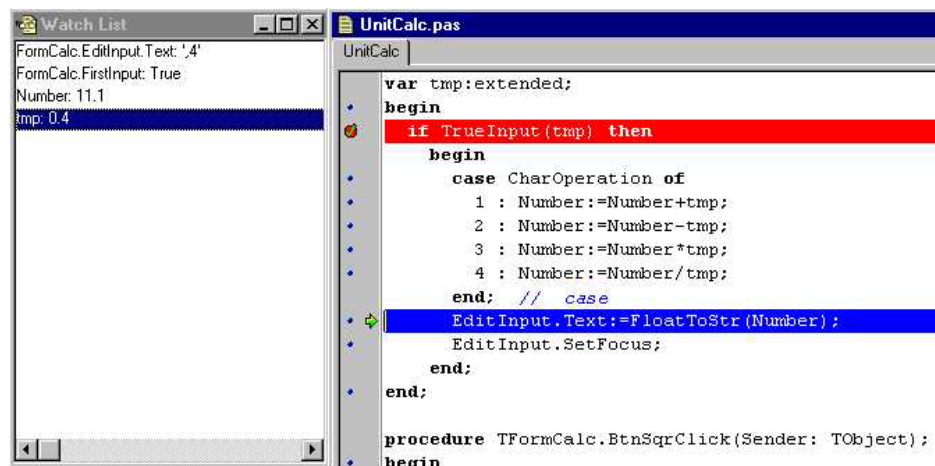


Рис. 1.12

диалоговое окно Evaluate/Modify модификации переменной: в верхней строке Expression – имя переменной; в средней Result – текущее значение переменной; в нижней New Value – ее новое значение.

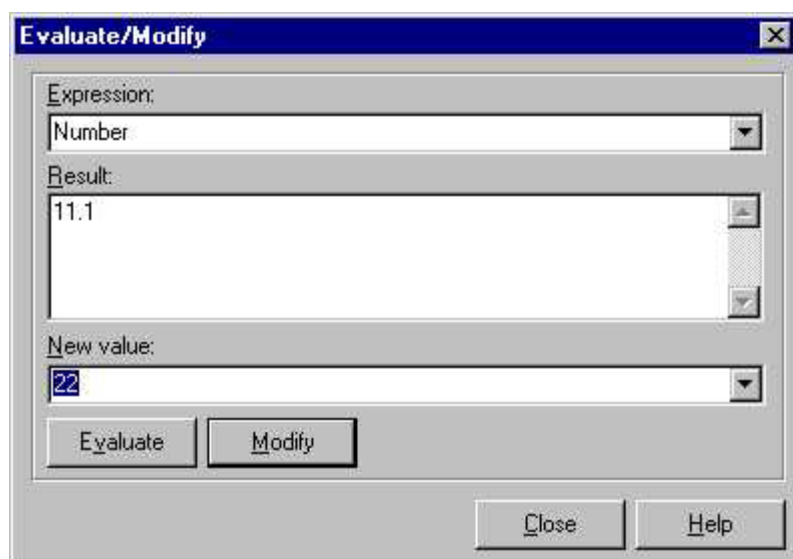


Рис. 1.13

Отладка программных средств позволяет не только выявить и устранить ошибки, но и значительно упростить и удешевить сопровождение программного продукта.

1.5. Макрогенераторы

При программировании на ассемблере часто бывает необходимо несколько раз повторять в программе одну и ту же последовательность команд. Ее можно оформить как подпрограмму и заменять в программе соответствующую последовательность команд обращением к подпрограмме. Недостатком такого способа является то, что при каждом обращении к подпрограмме должна выполняться команда перехода на подпрограмму и возврата из нее в основную программу. Если же подпрограмма заменяет короткую последовательность команд, например, две-пять команд, а количество обращений к ней велико, то затраты времени на организацию вызова подпрограммы могут быть сравнимы со временем ее работы.

Поэтому были разработаны так называемые **макроязыки** – включающие наряду с мнемоническими обозначениями машинных команд отдельные *макрокоманды* – языки программирования для

представления макроопределений, т.е. которые при трансляции заменяются группами команд машинного языка. Это сокращает запись исходной программы и упрощает программирование, т.к. исключаются записи часто повторяющихся фрагментов программы. В настоящее время все языки ассемблеров допускают использование макросредств, т.е. являются макроассемблерами (макросистемами). **Макросистема** – система программирования на языке ассемблер, включающая макроязык и макрогенератор

Макросредства ассемблера позволяют эффективно решить проблему многократного использования в программе одинаковых или почти одинаковых последовательностей команд.

МАКРОКОМАНДА, МАКРОВЫЗОВ И МАКРОРАСШИРЕНИЕ. Макрокоманда (макроопределение) – это сокращенная запись некоторой части программы в виде одной псевдокоманды, не входящей в систему команд компьютера. Пусть, например, программа содержит одинаковые последовательности команд для сохранения содержимого регистров R0, ... , R5 в стеке и их последующего восстановления. Как показано ниже, эти последовательности могут быть определены в виде макрокоманд SAVE и UNSAVE, соответственно:

.MACRO SAVE	;	Заголовок макроопределения
MOV R0, -(SP)	;	
MOV R1, -(SP)	;	
MOV R2, -(SP)	;	Тело
MOV R3, -(SP)	;	макроопределения
MOV R4, -(SP)	;	
MOV R5, -(SP)	;	
.ENDM	;	Конец макроопределения
.MACRO UNSAVE	;	Заголовок макроопределения
MOV (SP)+, R5	;	
MOV (SP)+, R4	;	
MOV (SP)+, R3	;	Тело
MOV (SP)+, R2	;	макроопределения
MOV (SP)+, R1	;	
MOV (SP)+, R0	;	
.ENDM	;	Конец макроопределения

Определение каждой макрокоманды начинается директивой ассемблера .MACRO и заканчивается директивой .ENDM, имеющими формат:

```
.MACRO name
.ENDM [name]
```

где name – имя макрокоманды.

С использованием определений макрокоманд SAVE и UNSAVE программа будет выглядеть следующим образом:

```
...
SAVE                ;      Вызов макрокоманды SAVE
CALL SUBR1           ;      Вызов подпрограммы SUBR1
UNSAVE              ;      Вызов макрокоманды UNSAVE
...
SAVE                ;      Вызов макрокоманды SAVE
CALL SUBR2           ;      Вызов подпрограммы SUBR1
UNSAVE              ;      Вызов макрокоманды UNSAVE
```

причем всякий раз, когда в поле операции встречаются макрокоманды SAVE и UNSAVE, они заменяются при трансляции телом соответствующей макрокоманды.

Использование имени макрокоманды в поле операции называется **макрывызовом**, а замена его при трансляции на тело макрокоманды – макрорасширением. **Макрорасширение** – последовательность предложений, порождаемая макрогенератором при обработке макрокоманды под управлением макроопределения и вставляемая в программу вместо макрокоманды.

Не следует путать макривызовы с вызовом подпрограммы. Принципиальная разница между ними состоит в том, что *макривызов* – это псевдокоманда, замещаемая во время трансляции телом соответствующей макрокоманды, а вызов подпрограммы – машинная команда, передающая управление на подпрограмму во время выполнения программы. В табл. 1.1 приведено сравнение макривызова и вызова подпрограммы.

Макрокоманда может быть определена в исходной программе пользователя либо в библиотеке макрокоманд, если она является служебной или представляет интерес для использования в других программах. В первом случае макрокоманда должна быть определена до момента ее макривызова, во втором необходимо указать имя макрокоманды в списке аргументов директивы .MCALL, но опять же до ее макривызова. Директива ассемблера .MCALL имеет формат:

.MCALL name1 [, name2,..., nameN]

где name1, name2, nameN – имена библиотечных макрокоманд.

Описанный выше способ определения макрокоманд может быть использован для уменьшения длины исходного текста программ, в которых повторяются абсолютно идентичные последовательности команд. Однако часто программы содержат похожие, но не полностью идентичные последовательности команд. Для случая схожих последовательностей команд в определение макрокоманды вводятся формальные параметры, а в макривызов – фактические (**макропараметры**). При

Таблица 1.1

Сравнение макровызова и вызова подпрограммы

Характеристика	Макровывоз	Вызов подпрограммы
Когда происходит вызов	При трансляции	При выполнении
Вставляется ли тело в объектную программу каждый раз, когда встречается вызов	Да	Нет
Сколько копий тела присутствует в объектной программе	По одной на каждый макровывоз	Одна
Остается ли в объектной программе после трансляции команда вызова	Нет	Да
Нужна ли команда возврата в исходную программу	Нет	Да

макрорасширении макрокоманды каждый формальный параметр в теле макрокоманды заменяется соответствующим ему фактическим параметром. При макровывозе фактические параметры макрокоманды записываются в поле операндов.

Соответствие между формальными и фактическими параметрами ассемблер устанавливает путем просмотра слева направо списка параметров макровывоза и директивы MACRO определения макрокоманды. Параметры должны быть отделены друг от друга символами разделителей (пробел или «,»). Появление в списке параметров двух подряд идущих разделителей будет восприниматься ассемблером как пустой параметр. При макрорасширении вместо соответствующего ему формального параметра в этом случае будут вставлены пробелы. Ассемблер выполняет замену формальных параметров на фактические в теле макрокоманды формально, проверяя лишь, чтобы получившаяся в результате строка была допустимой конструкцией языка.

Макрокоманды и макросы широко применяются и сейчас. Это обусловлено, во-первых, тем, что программы взаимодействия с аппаратными средствами ЭВМ требуют высокого быстродействия, которое достигается только при использовании низкоуровневого программирования, и, во-вторых, их широким использованием в современных пакетах прикладных программ (Word, Excel, Access, Delphi, C++ и др.) для обеспечения выполнения ряда часто повторяющихся последовательностей операций (часто они определяются конкретным пользователем программы). Рассмотрим использование макросов и макрокоманд на примере Microsoft Access.

Макросом называют набор из одной или более макрокоманд, выполняющих определенные операции, такие как открытие форм или печать отчетов. Макросы могут быть полезны для автоматизации часто выполняемых задач. Например, при нажатии пользователем кнопки можно запустить макрос, который распечатает отчет (рис. 1.14).

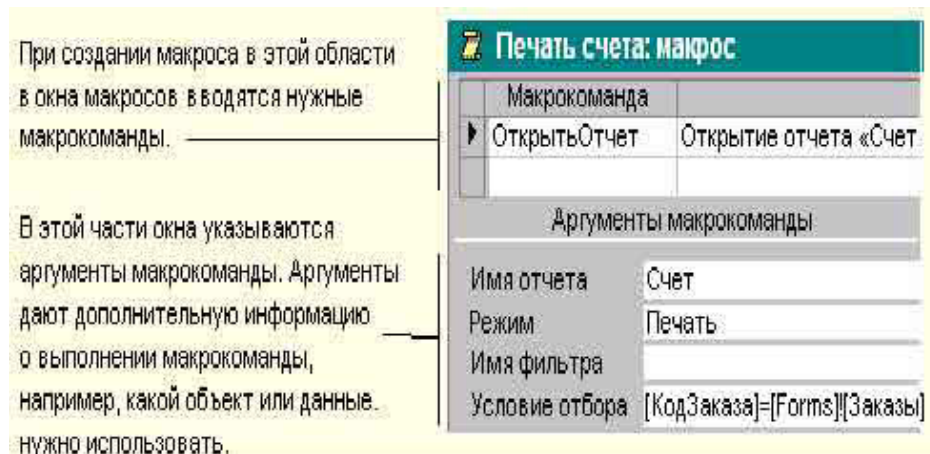


Рис. 1.14

Макрос может быть как собственно макросом, состоящим из последовательности макрокоманд, так и группой макросов. В некоторых случаях для решения, должна ли в запущенном макросе выполняться определенная макрокоманда, может применяться условное выражение.

Следующий макрос (рис. 1.15) состоит из серии (последовательности) макрокоманд. Эти макрокоманды выполняются каждый раз при запуске макроса. Для запуска макроса следует обратиться к имени макроса <Просмотр товаров>.

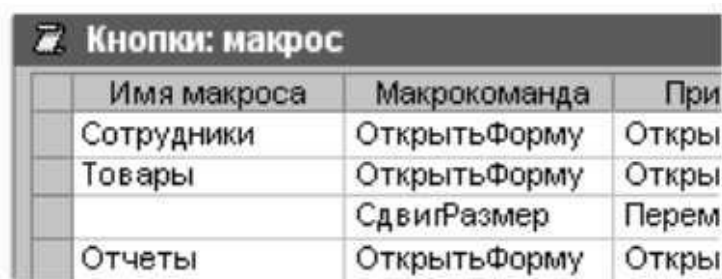
Просмотр товаров : макрос	
Макрокоманда	Примечание
	Связан с отчетом
Сигнал	Фиксация экрана до
ОткрытьФорму	Открытие формы
▶ СдвигРазмер	Перемещение спис

Рис. 1.15

При наличии большого числа макросов, объединение родственных макросов в группы может упростить управление базой данных. Для просмотра имен макросов для выбранной группы макросов достаточно в окне макроса в меню Вид выбрать команду Имена макросов.

Например, следующая группа макросов с именем <Кнопки> (рис. 1.16) состоит из трех родственных макросов: <Сотрудники>,

<Товары> и <Отчеты>. В каждом макросе содержится макрокоманда Открыть Форму (OpenForm), а в макросе <Товары>, кроме того, макрокоманда Сдвиг Размер (MoveSize).



Имя макроса	Макрокоманда	При
Сотрудники	ОткрытьФорму	Откры
Товары	ОткрытьФорму	Откры
	СдвигРазмер	Перем
Отчеты	ОткрытьФорму	Откры

Рис. 1.16

Имя в столбце Имя макроса определяет каждый макрос. При запуске макроса в группе макросов, выполняется макрокоманда в столбце Макрокоманда, а также все следующие макрокоманды, в которых столбец Имя макроса пуст. Для запуска макроса из группы макросов следует указать имя группы, а затем, через точку, имя макроса. В предыдущем примере для обращения к макросу <Сотрудники> в группе макросов <Кнопки> следовало использовать синтаксис – Кнопки.Сотрудники.

Существуют также условные макрокоманды. Для вывода столбца Условие следует в окне макроса в меню Вид выбрать команду Условия. Следующий макрос (рис. 1.17) запускает макрокоманды Сообщение (MsgBox) и ОстановитьМакрос (StopMacro) только в тех случаях, когда условие в столбце Условие истинно (когда поле <КодПоставщика> имеет значение Null).



Условие	Макрокоманда
IsNull([КодТовара])	Сообщение
	ОстановитьМакрос

Рис. 1.17

При запуске макроса выполнение макрокоманд начинается с первой строки макроса и продолжается до конца макроса, или, если макрос входит в группу макросов, до начала следующего макроса. Выполнение макроса может начинаться по команде пользователя, при вызове из другого макроса или процедуры обработки события, а также в ответ на

событие в форме, отчете или элементе управления. Например, можно назначить макрос на кнопку в форме, в результате чего макрос будет запускаться при нажатии кнопки. Допускается также создание специальной команды меню или кнопки панели инструментов, запускающей макрос, определение сочетания клавиш, нажатие которых запускает макрос, а также автоматический запуск макроса при открытии базы данных.

1.6. Системы управления базами данных

Современные информационные системы организационного управления предназначены оказывать помощь специалистам, руководителям, принимающим решения, в получении ими своевременной, достоверной и в необходимом количестве информации. Достигается это переходом на новую информационную технологию.

Новая информационная технология – это технология, основанная на:

- повсеместном применении ЭВМ и оргтехники;
- активном участии пользователей (непрофессионалов в области вычислительной техники и программировании) в информационном процессе;
- высоком уровне дружественного пользовательского интерфейса;
- широком использовании пакетов прикладных программ (ППП) общего и проблемного назначения;
- возможности для пользователя доступа к базам данных и программ, в том числе и удаленным, благодаря локальным и глобальным сетям ЭВМ;
- анализе ситуаций при выработке и принятии управленческих решений с помощью автоматизированных рабочих мест специалистов;
- применении систем искусственного интеллекта;
- внедрении экспертных систем;
- использовании телекоммуникации;
- создании геоинформационных систем и других технологий.

1.6.1. Автоматизированные банки данных

В настоящее время накоплен большой опыт разработки автоматизированных систем управления. Этот опыт говорит о том, что центральным техническим вопросом разработки информационных систем является организация, хранение и комплексное использование данных. В конечном счете это привело к созданию развитых средств управления данными, которые являются основой любой информационной системы, построенной на базе использования средств вычислительной техники.

Под *автоматизированным банком данных* понимается организационно-техническая система, представляющая совокупность баз данных пользователей, технических и программных средств формирования и ведения этих баз и коллектива специалистов, обеспечивающих функционирование системы.

В самом общем виде основные функции банка данных можно сформулировать следующим образом: адекватное информационное отображение предметной области (совокупности описания объектов реального мира и связей между ними, актуальных для данной прикладной деятельности), обеспечение хранения, обновления и выдачи необходимых данных пользователям. Составными частями любого банка данных являются база данных (БД), система управления базой данных (СУБД), администратор базы данных, прикладное программное обеспечение (рис. 1.18).



Рис. 1.18

Функционирование системы управления базой данных основано на введении двух уровней организации базы данных – логического и физического. Эти два уровня соответствуют двум аспектам организации данных: физическому с точки зрения хранения данных в памяти и логическому с точки зрения использования данных в прикладных приложениях.

Описание логических организаций баз данных определяет взгляд пользователей на организацию данных в системе, которые отображают

состояние некоторой предметной области. Необходимо отметить, что в общем случае структуры физической и логической организации данных могут не совпадать. Формальное описание логической организации данных иногда называют моделью данных или схемой.

Говоря о физической организации, необходимо отметить, что существует много различных способов организации данных в запоминающей среде, с помощью которых можно обеспечить соответствие некоторой модели.

Наиболее общее представление о базе данных заключается в следующем: *база данных (data base)* – это совокупность хранимых во внешней памяти ЭВМ большого объема данных; база данных является «интегрированной», т.е. представляет собой комплекс взаимосвязанных данных, предназначенный для обеспечения информационных нужд различных пользователей, каждый из которых имеет отношение к отдельным, возможно, совместно используемым частям данных; работа с базой данных может осуществляться либо в пакетном режиме, либо с удаленных терминалов в режиме реального времени. База данных должна быть непротиворечива, минимально избыточна и целостна. Структура системы баз данных представлена на рис. 1.19.

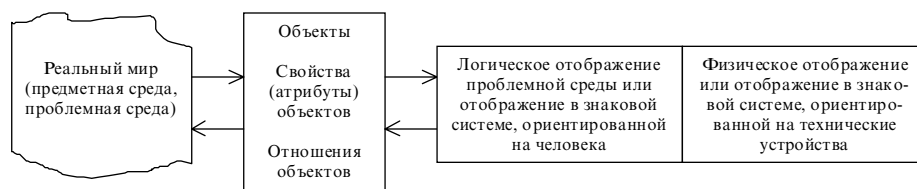


Рис. 1.19

Таким образом, *база данных* – это совокупность хранимых в памяти ЭВМ и специальным образом организованных взаимосвязанных данных, отображающих состояние предметной области. База данных также предназначена для обеспечения информационных нужд определенных пользователей.

Создание единой базы данных о предметной области сложно и в настоящее время практически нереализуемо, хотя бы из-за недостаточного объема памяти современных ЭВМ. На практике большинство баз данных проектируется для ограниченного числа приложений. На одной ЭВМ, как правило, создается несколько различных баз данных. Со временем некоторые базы данных, предназначенные для выполнения родственных функций, могут объединиться, если это будет способствовать повышению производительности всего вычислительного комплекса.

Создание баз данных обеспечивает интеграцию данных и возможность централизованного управления данными.

1.6.2. Объекты и отношения объектов

Любая информационная система должна отображать те или иные стороны окружающего нас реального мира или, как иногда говорят, проблемной или предметной области. Мы воспринимаем окружающий мир состоящим из объектов, которые человек, по совокупности определенных достаточно устойчивых свойств, группирует в наборы (классы) объектов, которым он присваивает имя. Например, в реальном мире есть конкретные собаки, но нет собаки «вообще». Понятие «собаки» описывает целый класс в каком-то смысле однородных реальных объектов.

Проблемная среда изменяется со временем, что выражается в изменении свойств объектов, возникновении новых и исчезновении старых объектов. Эти изменения происходят в результате событий. Временная последовательность событий образует процесс.

Всякая информационная система имеет дело не с самими объектами, как реальными сущностями, а с их знаковыми отображениями-идентификаторами. Главная функция знака-идентификатора – отличить объект в группе однородных объектов. Идентификатор объекта, вообще говоря, может не нести никакой информации о свойствах объекта или, что то же самое, об его принадлежности к тому или иному классу.

Например, 11591 – «табельный номер служащего» – является числовым идентификатором. Этот идентификатор не описывает свойства, их приходится задавать дополнительно.

Более полно объект описывается записью об объекте, которая обычно состоит из идентификатора объекта-знака, позволяющего отличить один объект от другого среди однородных объектов, и идентификаторов (значений) свойств (атрибутов). Например, запись о служащем некоторой организации имеет табельный номер служащего в качестве идентификатора и такие элементы данных, как должность, заработная плата, льготы и т.д., рассматриваемые как идентификаторы (значения) свойств служащего.

Следует подчеркнуть, что понятие объекта и свойства относительны. Если речь идет о служащем, то естественно понимать должность как свойство служащего. Но если речь идет о должности, например, в смысле должностных инструкций, то уже сама должность выступает в качестве объекта, который может иметь свойства. В частности, в определенном контексте табельный номер служащего может рассматриваться как свойство должности.

Поэтому при информационном отображении предметных сред (рис. 1.20) можно (а иногда и нужно) говорить не об объектах и их свойствах, а об отношениях объектов, ибо в этом случае все идентификаторы в записи можно рассматривать симметрично, а не в ориентации на один специально выделенный объект. Как увидим позднее, это соответствует так называемой реляционной точке зрения на базу данных.

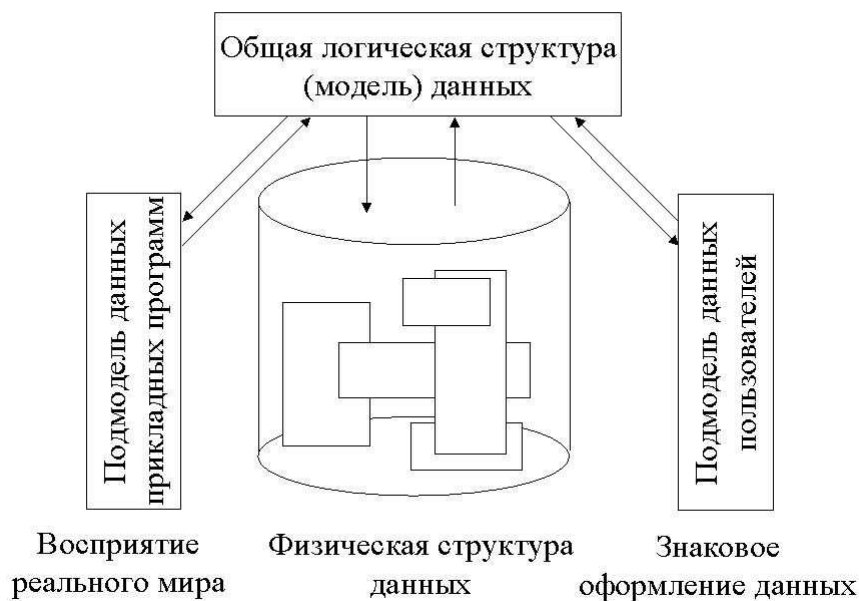


Рис. 1.20

При информационном отображении реального мира весьма важно, в каких количественных пропорциях могут осуществляться отношения объектов. Четкое понимание того, к какой категории относится отношение объектов, позволяет сделать заключение о возможном характере связи между соответствующими данными. Фиксация этой стороны при информационном отображении предметной области определяет одну из сторон модели данных. Важно подчеркнуть, что характер отношений одних и тех же объектов не есть нечто застывшее. Он может измениться, и тогда изменится характер связей между элементами данных, который может оказать существенное влияние на структуру банка данных, как логическую, так и физическую. Усложнение характера связей между данными делает более сложными программы их обработки.

1.6.3. Данные

Информация об объекте или отношениях объектов, выраженная в знаковой форме, образует данные. Эти данные могут быть восприняты человеком или каким-либо техническим устройством и соответствующим образом интерпретированы.

Характерной особенностью данных является то, что их можно переводить из одной знаковой системы в другую (перекодировка) без потери информации. Это существенное свойство знакового отображения позволяет описывать реальную предметную ситуацию в различных

системах знаков, ориентированных на воспринимающего. При построении банков данных стало уже традиционным говорить о логическом отображении, ориентированном на человека, и о физическом отображении, ориентированном на устройства долговременной памяти.

На рис. 1.20 воспроизведены основные понятия, связанные с отображением некоторой предметной области в банке данных.

Следует отметить, что знаки сами по себе не образуют данных, несущих информацию о предметной области. В простейшем случае знаки должны быть структурно оформлены в виде фиксированной последовательности-записи, а каждое поле записи (в которое помещается знак) должно иметь интерпретацию с точки зрения предметной области, для которой создается банк данных. Например, если знаки образуются из букв русского алфавита по правилам образования слов русского языка, будучи взяты сами по себе, они несут только информацию о правилах образования знаков – синтаксическую информацию. И в этом смысле их можно рассматривать как данные.

С точки зрения отображения предметной области для нас представляет интерес семантическая информация, а именно, как определенные знаки связаны с объектами предметной области и их отношениями. Самый простейший способ реализовать эту связь – это придать определенное содержание (смысл) полю записи. Например, если поле интерпретируется как «должность служащего», то любые знаки, помещенные в это поле, будут пониматься как конкретные идентификаторы различных должностей, и в этом смысле мы их рассматриваем как данные о предметной области.

Целесообразно кратко рассмотреть прагматический аспект знакового отображения. Если определены структуры записей обо всех объектах предметной области и их отношениях, то будем говорить, что задана модель данных предметной области.

Если предметная область обширна, например, производственная деятельность предприятия, то и ее модель данных будет достаточно велика. На предприятии практически нет человека, который представляет производственную деятельность во всей ее детализации, а это значит, что модель данных в целом никому из управляющего состава предприятия не нужна. Сама эта проблема (построение общей модели) возникла только в связи с разработкой и эксплуатацией автоматизированных информационных систем. Для каждого конкретного отдела или звена управления производственной деятельности характерна своя сфера информационных интересов, которая тоже может быть описана своей моделью данных.

По отношению к общей модели данных рассмотрим ее как подмодель. В общем случае подмодель данных – это не простое механическое усечение модели. Подмодель данных конкретного пользования может быть связана со всей моделью данных весьма сложными структурными преобразованиями. Понятие подмодели характеризует прагматический аспект знакового отображения.

1.6.4. Модель и подмодель данных

Банк данных хранит информацию об объектах реального мира и отношениях между этими объектами через данные и связи между этими данными. Прежде чем говорить о размещении данных и связей между ними на устройствах памяти, необходимо представить взаимосвязь данных на логическом уровне, создав своеобразную модель данных.

Эта модель данных должна быть четко определена, для чего в системе управления базами необходимо представить средства (язык) для ее описания. Основное назначение *модели данных* состоит в том, чтобы дать возможность представить в целом информационную картину без отвлекающих деталей, связанных с особенностями хранения. Она является инструментом, с помощью которого разрабатывается стратегия получения любых данных, хранящихся в банке.

Термином *подмодель* определяют описание данных, используемое при прикладном программировании. На основе единой модели можно составить множество различных подмоделей. Теоретически они могут сильно отличаться от общей модели данных. Принципиально здесь то, что можно указать алгоритм, с помощью которого данные со структурой подмодели получаются из базы данных, построенной в соответствии с моделью.

Рассмотрим *иерархическую, сетевую и реляционную модели данных*. Можно считать, что в некотором смысле эта последовательность отражает прогресс в понимании проблем обработки данных. Однако начнем рассмотрение их все-таки с сетевой модели, так как она наиболее наглядно отражает отношения объектов проблемной области.

СЕТЕВАЯ МОДЕЛЬ ДАННЫХ. Отношения объектов реального мира всегда могут быть представлены в виде некоторой сети. Это представление рисует довольно наглядную картину реальной действительности и, кроме того, претендует на то, что может быть естественным образом отражено в долговременной памяти вычислительной (информационной) системы. Пример сетевой модели данных приведен на рис. 1.21, из которого видно, что если от элемента В есть ссылка на элемент А, то возможен выбор элемента А.

Каждый узел сети соответствует элементу данных, отображающему группу однородных объектов реального мира. Тогда в реальной сети или, как говорят, экземпляре сети в каждом узле будет находиться идентификатор соответствующего объекта, например, шифр детали.

На самом деле представлять и рисовать сеть на таком уровне детализации достаточно трудно. Поэтому в вершинах сети обычно стоят целые записи, состоящие из совокупности идентификаторов. Например, в вершине сети, имеющей имя деталь, фактически стоит запись (сегмент), состоящая из полей шифр детали, наименование детали и масса детали. На каком основании объединять различные элементы данных в записи (сегменты) – это предмет особого разговора.

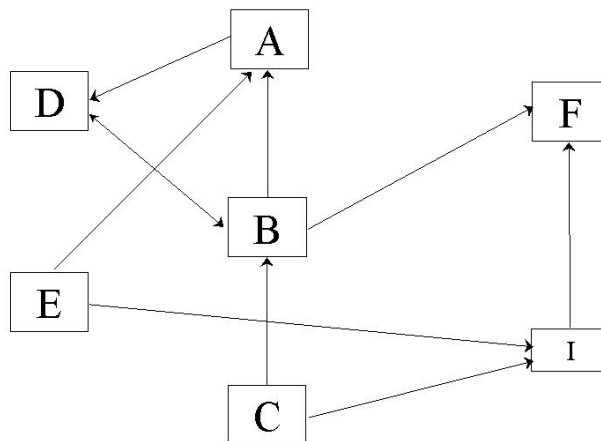


Рис. 1.21

ИЕРАРХИЧЕСКАЯ МОДЕЛЬ ДАННЫХ. Более удобное представление данных достигается за счет увеличения информационной точности на уровне модели данных. Шагом на пути к увеличению информационной избыточности является переход от сетевого к иерархическому представлению данных. Сеть можно представить в виде совокупности деревьев. Для этого в иерархических структурах требуется повторить и несколько преобразовать некоторые вершины сети.

Если на уровне сети, не вдаваясь в подробности физического хранения данных, база данных представляется в виде сложной объемной паутины, то на уровне иерархической модели базы данных представляются в виде совокупности отдельных древовидных структур, в корнях которых стоят идентификаторы объектов, а на последующих ярусах раскрываются свойства этих объектов (из рис. 1.22 видно, что для доступа к элементу A12, нужно сначала найти в БД узел А, затем спуститься к узлу A1 и только после этого получить доступ к элементу A12).

Таким образом, по крайней мере на уровне логического представления, некоторые отношения и связи реального мира имитируются с информационной избыточностью.

Это еще не значит, что информационная избыточность будет реальной, т.е. будет иметь место и на уровне физического отображения на долговременную память. Такое видение мира более удобно с точки зрения обработки данных.

Следовательно, сетевые модели данных могут быть разложены на иерархические структуры. Однако следует отметить, что модель данных, представленная как совокупность нескольких деревьев, не обладает

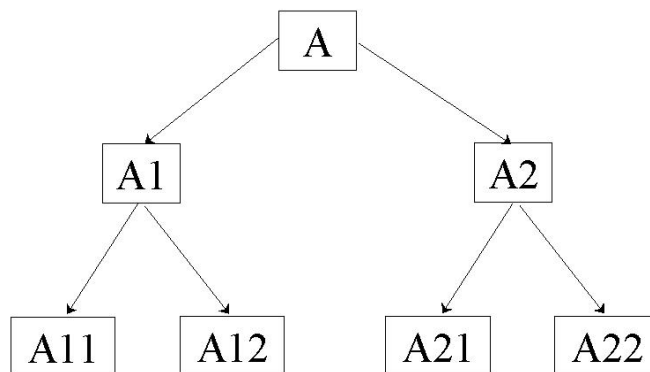


Рис. 1.22

наглядностью, так как не создается впечатления некоторой взаимосвязанной системы данных. Вообще говоря, одновременно с использованием деревьев для описания модели данных неплохо представлять себе модель данных в виде сети, с тем чтобы ясно понимать, как с ней взаимодействуют иерархические структуры.


Сетевая модель, если она не очень громоздка, позволяет графически представить взаимодействие объектов, отображенных в базе данных. Это всегда полезно, даже если затем для описания используются иные структуры.

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ. Наиболее абстрактной моделью является реляционная модель данных. Абстрактна она в том смысле, что в значительной степени ориентирована на интересы пользователя (программиста) и совершенно не несет в себе черт реального отображения на физическую память. Эта модель исторически возникла позже других, и ее появление оправдывается тем, что по мере усложнения информационных систем и прогресса в устройствах долговременной памяти поддержание математического обеспечения, несущего в себе черты «суеверия представления», обходится очень дорого.

Реляционная модель получается путем дальнейшей формализации иерархической модели. В этой модели все связи между объектами задаются путем явной фиксации идентификаторов объектов в записях.

На первый взгляд реляционная модель может быть представлена в виде однородных таблиц (отношений), которые напоминают стандартные последовательные файлы. Однако это очень упрощенная точка зрения. Дело в том, что последовательный файл предполагает определенную упорядоченность и обработку в соответствии с этой упорядоченностью, или, иными словами, предполагается вход со стороны одного поля (ключа сортировки). На рис. 1.23 представлен пример такой модели: слева таблица «Товары», справа связанная с

ней по полю «товар» таблица «Отпуск товара», так называемое отношение один-ко-многим. (Отметим, что существуют также отношения «один-к-одному» и «многие-ко-многим»).



Товар	Ед.изм.	Цена ед.
Сахар	кг	5000
Макароны	кг	7000
Куры	кг	10000
Фанта	бут.1 л	6000

Товар	Дата	Кол-во, ед.
Сахар	10.01.97	100
Сахар	12.01.97	200
Сахар	14.01.97	50
Макароны	10.01.97	1000
Макароны	11.01.97	500
Фанта	10.01.97	2000
Фанта	12.01.97	3000

Рис. 1.23

Существенное отличие реляционной модели от обыкновенного последовательного файла заключается в том, что все столбцы в таблице с точки зрения входа предполагаются эквивалентными. Именно это свойство делает эту модель весьма мощной и делает невозможным отображение ее на память в виде последовательного массива данных.

Информационная избыточность, по крайней мере на логическом уровне, максимальная. Однако повторим еще раз: это не предполагает реальной информационной избыточности при отображении данных на физическую память.

На физическом уровне, например, может быть абсолютно неизбыточной сеть или сеть с перекрестными связями. Задача системы управления базой данных заключается в том, чтобы перевести запрос, сформулированный в терминах реляционной модели, в последовательность команд, осуществляющих поиск по сети. Таким образом, реляционная модель предполагает очень высокую степень «интеллектуальности» систем управления базой данных.

1.6.5. Функционирование СУБД

СУБД обеспечивает работу прикладных программ с базой данных. Одним из важнейших назначений СУБД является обеспечение независимости данных. Под этим термином понимается независимость данных и использующих их прикладных программ друг от друга в том смысле, что изменение одних не приводит к изменению других. Необходимо также отметить такие возможности СУБД, как обеспечение защиты и секретности данных, восстановление баз данных после сбоев, ведение учета работы с базами данных. Однако это является неполным перечнем того, что должна осуществлять СУБД для обеспечения интерфейса пользователей с базами данных и жизнеспособности всего автоматизированного банка данных.

Система управления данными имеет набор средств, которые обеспечивают определенные способы доступа к данным. Наиболее общими операциями, которые выполняются средствами СУБД, являются операции поиска, исправления, добавления и удаления данных. Необходимо отметить, что операция поиска является главной среди указанных.

Степень реализации принципа независимости данных определяет гибкость системы управления базами данных. Учет особенностей обработки данных в какой-либо предметной области позволяет спроектировать специализированные СУБД, ориентированные на применение в АСУ предприятий с дискретным характером производства.

Существуют и универсальные системы управления базами данными, используемые для различных приложений. При настройке универсальных СУБД для конкретных приложений они должны обладать соответствующими средствами. Процесс настройки СУБД на конкретную область применения называется генерацией системы. К универсальным системам управления базами данных относятся, например, системы dBase, Paradox, Microsoft Access, Oracle.

Работа системы управления базами данных связана с использованием трех уровней описаний данных: описание физической организации данных, описание общей логической структуры данных, серии описаний подмоделей данных прикладной программы.

Представим последовательность действий, которые должна выполнить система управления базой данных, в процессе формирования записи соответствующей подмодели данных прикладной программы. Наиболее важные действия приведены на рис. 1.24.

1. Прикладная программа (ПП) формирует запрос на чтение записи. Если обращение осуществляется к конкретной записи, то формируется значение ключа записи. Если записи обрабатываются последовательно, то ключей не требуется.
2. Система управления базой данных на основании описания подмодели данных для данной прикладной программы исследует вопрос о правомочности обращения к данным из программы.
3. На основе общей модели данных СУБД привязывает подмодель данных к модели и определяет, какие элементы данных необходимы.
4. На основе описания физической организации данных определяется, какие физические записи следует считать в системные буферы, чтобы сформировать затем требуемые данные.
5. Операционной системе выдается задание на чтение требуемых физических данных.
6. Работают программы, реализующие методы доступа операционной системы.
7. Из внешних запоминающих устройств запрошенные физические записи перемещаются в системные буферы.

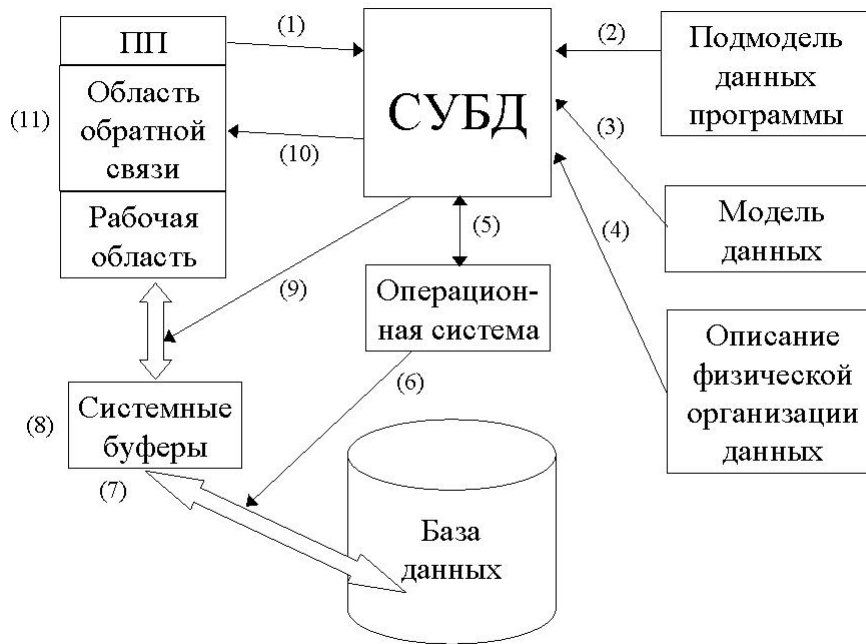


Рис. 1.24

8. СУБД на основе описания общей модели данных, а также описания подмоделей программы формирует логическую запись в соответствии с подмоделью. При этом реализуется необходимое преобразование данных, вызванное разным определением данных в модели и подмодели.
9. Данные из системных буферов СУБД передаются в рабочую область прикладной программы.
10. Система управления базой данных формирует и передает прикладной программе код возврата или информацию о своей работе в процессе обслуживания ее запроса.
11. Прикладная программа приступает к функциональной обработке переданных ей данных. Примерно аналогичные процессы происходят при попытке прикладной программы записать данные в базу данных. В принципе СУБД должна осуществлять обратные преобразования по отношению к чтению данных в соответствии с подмоделью, моделью, физической организацией.

Однако на практике на подмодели для записи данных наложено значительно больше ограничений, чем на подмодели, предназначенные для чтения данных.

Следует иметь в виду, что СУБД ведет одновременную обработку нескольких прикладных программ, которые могут иметь уникальные

подмодели данных. Отсюда понятно, что взаимодействие с системными буферами должно обеспечить одновременную работу с различными данными. От того, насколько часто меняется содержание системных буферов, в значительной степени зависит общее быстродействие системы.

Специалист по общей организации банка данных имеет дело со всеми подмоделями данных, используемыми в ПП, общей моделью данных, общей конфигурацией системы управления базами данных и операционной системы.

Специалист по физической организации банка данных в долговременной памяти заботится об эффективной организации данных в памяти на основе изучения реальной статистики запросов. Все его действия по оптимизации физической организации БД не должны влиять на общую логическую модель данных – в этом заключается принцип физической независимости. Кроме того, он решает все вопросы, связанные с взаимодействием системы управления базами данных и операционной системы.

В настоящее время большинство современных систем управления базами данных наряду с обслуживанием прикладных программ в режиме пакета осуществляют еще обработку запросов с терминалов.

ОРГАНИЗАЦИЯ ПОИСКА ДАННЫХ. Записи логического файла идентифицируются с помощью уникальной группы символов – *ключа*. Обычно ключом является поле или совокупность полей фиксированной длины. В общем случае в качестве ключа может выступать любое поле записи. Каждому значению ключа может соответствовать одна или несколько записей файла. Ключ, каждому значению которого соответствует одна и только одна запись файла, называется *первичным*, или *основным*, ключом. Логические записи файла могут иметь несколько первичных ключей.

Каждой записи при ее хранении в памяти соответствует вполне определенное место, задаваемое адресом. Способ, задающий соответствие между основным ключом записи и ее адресами памяти, называется *способом адресации*. Основную проблему при адресации файла можно сформулировать следующим образом: как по основному ключу определить местоположение записи с данным ключом? Существует много различных способов адресации файлов, которые можно отнести к следующим трем основным группам: способы последовательной, индексной и прямой адресации. Используются также различные комбинации этих основных способов адресации.

Способы *последовательной адресации* характеризуются тем, что логические записи занимают непрерывный участок памяти и порядок их расположения определяется не значением ключа, а номером следования этого ключа в заданной последовательности ключей логической записи, т.е. не существует однозначного соответствия между значением одного ключа и адресов памяти.

При *индексной адресации* соответствие между основным ключом записи и ее адресом в памяти задается с помощью таблицы или иерархии таблиц, т.е. индекса. Способы *прямой адресации* характеризуются наличием некоторого алгоритма преобразования значения ключа записи непосредственно в адрес ее расположения во внешней памяти.

В зависимости от способов адресации файла могут использоваться различные способы локализации (поиска) записи с заданным знанием ключа. При последовательной адресации поиск записи может осуществляться только путем просмотра (сканирования) файла с проверкой ключей записи.

Существует много различных способов поиска записей, например блочный и дихотомический поиск. Наиболее простым способом является последовательное сканирование с проверкой ключа каждой записи.

При использовании способов индексной адресации поиск записи осуществляется путем предварительного просмотра индекса. В результате поиска в индексе сокращается объем просматриваемой памяти, занимаемой непосредственно записями файла. Сам индекс и его просмотр могут быть организованы различными способами. Поиск записей файлов, использующих прямую адресацию, осуществляется в соответствии с выбранным алгоритмом преобразования значения ключа записи в ее адрес.

АДМИНИСТРАТОР БАЗЫ ДАННЫХ. Администратора базы данных можно рассматривать как необходимый структурный элемент автоматизированного банка данных, т.е. банк данных включает в себя не только данные, программы и оборудование, но еще и персонал. Администратору базы данных отводится важная роль – ответственность за общее управление системой баз данных.

В обязанности *администратора БД* входит следующее:

Определение информационного содержания базы данных. Администратор базы данных с учетом запросов пользователей к базе принимает решение о том, какая информация должна содержаться в базе данных, т.е. определяет информационное содержание базы данных и задает их общую логическую организацию, так называемую модель данных.

Определение структуры памяти и стратегии доступа. Администратор базы должен решить, каким образом данные представляются в базе данных, т.е. разработать физическую организацию данных.

Взаимодействие с пользователем. Администратор базы данных – это лицо (или группа лиц), которое имеет глобальное представление об организации данных в системе и несет ответственность за их сохранность. Пользователи при участии администратора имеют возможность корректно определить собственный «взгляд» на базу данных, что выражается в задании подмодели данных.

Определение стратегии отказа и восстановления. Работа автоматизированной системы, использующей банк данных, существенно зависит от его успешного функционирования. В случае повреждения по какой-либо причине всей базы данных или ее некоторой части необходимо предусмотреть возможность восстановления данных с минимальной задержкой и влиянием на сохранившуюся часть базы данных. Администратор должен определить и реализовать соответствующую стратегию восстановления.

Модернизация и эффективность работы базы данных. Администратор базы данных ответствен за такую организацию системы, которая обеспечит максимальную эффективность ее функционирования, а также за выполнение всех модернизаций базы данных, направленных на более полное удовлетворение требований пользователей.

Для выполнения своих функций администратор базы данных использует набор вспомогательных программ. Эти программы составляют существенную часть системы управления базами данных. К ним относятся, например, программы ведения системного журнала, хранящего сведения о каждом обращении в базу данных, программы восстановления базы данных и программы анализа статистики использования данных.

На практике администратор БД – это чаще всего не один человек, а группа лиц, так как решаемый круг вопросов слишком широк для компетенции одного человека. Они несут ответственность за функционирование интегрированной БД, имеют полномочия по корректировке БД, отвечают как за целостность данных, так и за защиту их от несанкционированного доступа и надежность системы в целом.

Повышение требований к оперативности информационного обмена и управления, а следовательно, к срочности обработки информации привело к созданию многоуровневых систем организационного управления объектами, какими являются, например, банковские, налоговые, снабженческие, статистические и другие службы. Их информационное обеспечение поддерживают сети автоматизированных банков данных, которые строятся с учетом организационно-функциональной структуры соответствующего многоуровневого экономического объекта, машинного ведения информационных массивов. Эту проблему в новых информационных технологиях решают распределенные системы обработки данных с использованием каналов связи для обмена информацией между базами данных различных уровней. За счет усложнения программных средств управления базами данных повышаются скорости, обеспечиваются защита и достоверность информации при выполнении экономических расчетов и выработке управленческих решений.

1.7. CASE-системы

1.7.1. Понятие CASE-систем

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем (ИС), создаваемых в различных областях экономики. Современные крупные проекты ИС характеризуются, как правило, следующими особенностями:

- сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;

- наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);

- отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем;

- необходимость интеграции существующих и вновь разрабатываемых приложений;

- функционирование в неоднородной среде на нескольких аппаратных платформах;

- разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;

- существенная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков и масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИС.

Для успешной реализации проекта объект проектирования (ИС) должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем.

В 70-х и 80-х годах при разработке ИС достаточно широко применялась структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания ИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей ИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако широкое применение этой методологии и следование ее рекомендациям при разработке конкретных ИС встречалось достаточно редко, поскольку при неавтоматизированной (ручной) разработке это практически невозможно. Действительно, вручную очень трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость, и тем более изменить. Если все же удастся создать строгую систему проектных документов, то ее переработка при появлении серьезных изменений практически неосуществима. Ручная разработка обычно порождала следующие проблемы:

- неадекватная спецификация требований;
- неспособность обнаруживать ошибки в проектных решениях;
- низкое качество документации, снижающее эксплуатационные качества;
- затяжной цикл и неудовлетворительные результаты тестирования.

С другой стороны, разработчики ИС исторически всегда стояли последними в ряду тех, кто использовал компьютерные технологии для повышения качества, надежности и производительности в своей собственной работе (феномен “сапожника без сапог”).

Перечисленные факторы способствовали появлению программно-технологических средств специального класса – *CASE-средств*, реализующих *CASE-технологию* создания и сопровождения ИС. Термин CASE (Computer Aided Software Engineering – разработка ПО с помощью компьютера) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином *CASE-средства* понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением

языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т.д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;

широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;

внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Согласно обзору передовых технологий (Survey of Advanced Technology), составленному фирмой Systems Development Inc. в 1996 г. по результатам анкетирования более 1000 американских фирм, CASE-технология в настоящее время попала в разряд наиболее стабильных информационных технологий (ее использовала половина всех опрошенных пользователей более чем в трети своих проектов, из них 85% завершились успешно). Однако, несмотря на все потенциальные возможности CASE-средств, существует множество примеров их неудачного внедрения, в результате которых CASE-средства становятся "полочным" ПО (shelfware).

Успешное внедрение CASE-средств должно обеспечить такие выгоды, как:

- высокий уровень технологической поддержки процессов разработки и сопровождения ПО;
- положительное воздействие на некоторые или все из перечисленных факторов: производительность, качество продукции, соблюдение стандартов, документирование;
- приемлемый уровень отдачи от инвестиций в CASE-средства.

Таким образом, CASE-средства позволяют увеличить эффективность разработки ПО, но вместе с тем требуют нового подхода к их разработке и сопровождению на протяжении всего жизненного цикла.

1.7.2. Модели жизненного цикла ПО

Стандарт ISO/IEC 12207 не предлагает конкретную модель жизненного цикла (ЖЦ) и методы разработки ПО (под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ). Модель ЖЦ зависит от специфики ИС и специфики условий, в которых последняя создается и функционирует. Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки. Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

К настоящему времени наибольшее распространение получили следующие две основные модели ЖЦ:

- каскадная модель (1970-1985 гг.);
- спиральная модель (1986-1990 гг.).

В изначально существовавших однородных ИС каждое приложение представляло собой единое целое. Для разработки такого типа приложений применялся каскадный способ. Его основной характеристикой является разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем (рис. 1.25). Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

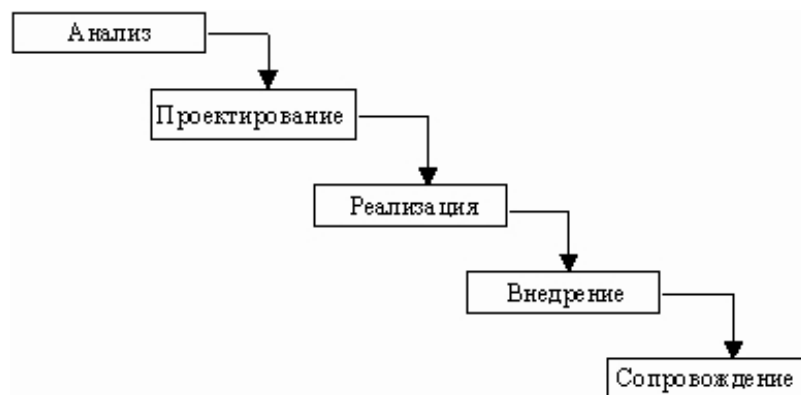


Рис. 1.25

Положительные стороны применения каскадного подхода заключаются в следующем:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;

- выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении ИС, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования, с тем чтобы предоставить разработчикам свободу реализовать их как можно лучше с технической точки зрения. В эту категорию попадают сложные расчетные системы, системы реального времени и другие подобные задачи. Однако в процессе использования этого подхода обнаружился ряд его недостатков, вызванных прежде всего тем, что реальный процесс создания ПО никогда полностью не укладывался в такую жесткую схему. В процессе создания ПО постоянно возникала потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ПО принимал следующий вид (рис. 1.26):

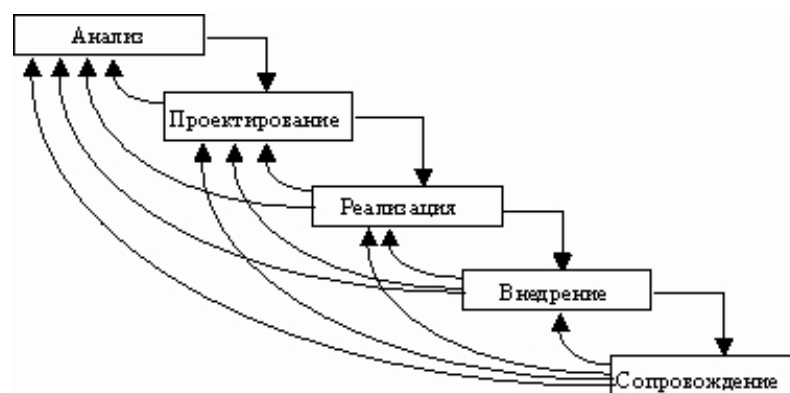


Рис. 1.26

Основным недостатком каскадного подхода является существенное запаздывание с получением результатов. Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ, требования к ИС “заморожены” в виде технического задания на все время ее создания. Таким образом, пользователи могут внести свои замечания только после того, как работа над системой будет полностью завершена. В случае неточного изложения требований или их изменения в течение длительного периода создания ПО, пользователи получают систему, не удовлетворяющую их потребностям. Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением.

Для преодоления перечисленных проблем была предложена спиральная модель ЖЦ (рис. 1.27), делающая упор на начальные этапы ЖЦ: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

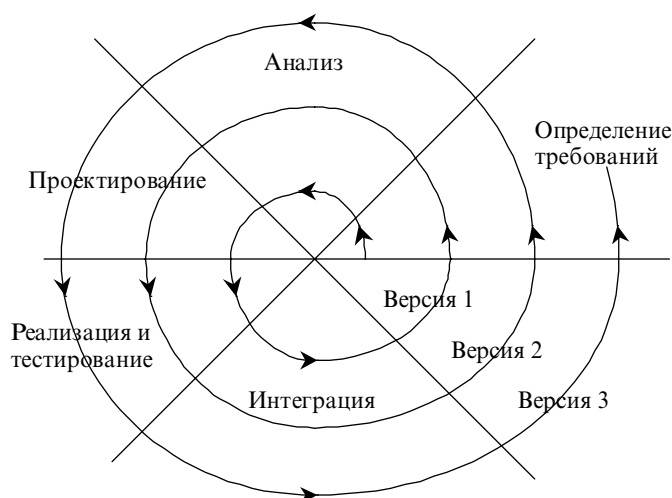


Рис. 1.27

Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем. При итеративном способе разработки недостающую работу можно будет выполнить на следующей итерации. Главная же задача – как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального цикла – определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

1.7.3. CASE-средства.

Общая характеристика и классификация

Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО.

Наиболее трудоемкими этапами разработки ИС являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

В разряд CASE-средств попадают как относительно дешевые системы для персональных компьютеров с весьма ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла ПО и обладающее следующими основными характерными особенностями:

- мощные графические средства для описания и документирования ИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности;

- интеграция отдельных компонентов CASE-средств, обеспечивающая управляемость процессом разработки ИС;

- использование специальным образом организованного хранилища проектных метаданных (репозитория).

Интегрированное CASE-средство (или комплекс средств, поддерживающих полный ЖЦ ПО) содержит следующие компоненты:

- репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;

- графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм, образующих модели ИС;

средства разработки приложений, включая языки 4GL и генераторы кодов;

средства конфигурационного управления;

средства документирования;

средства тестирования;

средства управления проектом;

средства реинжиниринга.

Все современные CASE-средства могут быть классифицированы в основном по типам и категориям. Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы ЖЦ. Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает отдельные локальные средства, решающие небольшие автономные задачи, набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла ИС и полностью интегрированные средства, поддерживающие весь ЖЦ ИС и связанные общим репозиторием. Помимо этого, CASE-средства можно классифицировать по следующим признакам:

применяемым методологиям и моделям систем и БД;

степени интегрированности с СУБД;

доступным платформам.

Классификация по типам в основном совпадает с компонентным составом CASE-средств и включает следующие основные типы:

средства анализа (Upper CASE), предназначенные для построения и анализа моделей предметной области;

средства анализа и проектирования (Middle CASE), поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций. Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;

средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД;

средства разработки приложений;

средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке C++.

Вспомогательные типы включают:

средства планирования и управления проектом;

средства конфигурационного управления;

средства тестирования;

средства документирования.

На сегодняшний день Российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами:

Vantage Team Builder (Westmount I-CASE); Designer/2000; Silverrun; ERwin+BPwin; S-Designer; CASE.Аналитик.

1.7.4. Оценка и выбор CASE-средств

Модель процесса оценки и выбора, рассматриваемая ниже (рис. 1.28), описывает наиболее общую ситуацию оценки и выбора, а также показывает зависимость между ними. Как можно видеть, оценка и выбор могут выполняться независимо друг от друга или вместе, каждый из этих процессов требует применения определенных критериев.

Процесс оценки и выбора может преследовать несколько целей, включая одну или более из следующих:

- 1) оценка нескольких CASE-средств и выбор одного или более из них;
- 2) оценка одного или более CASE-средств и сохранение результатов для последующего использования;
- 3) выбор одного или более CASE-средств с использованием результатов предыдущих оценок.

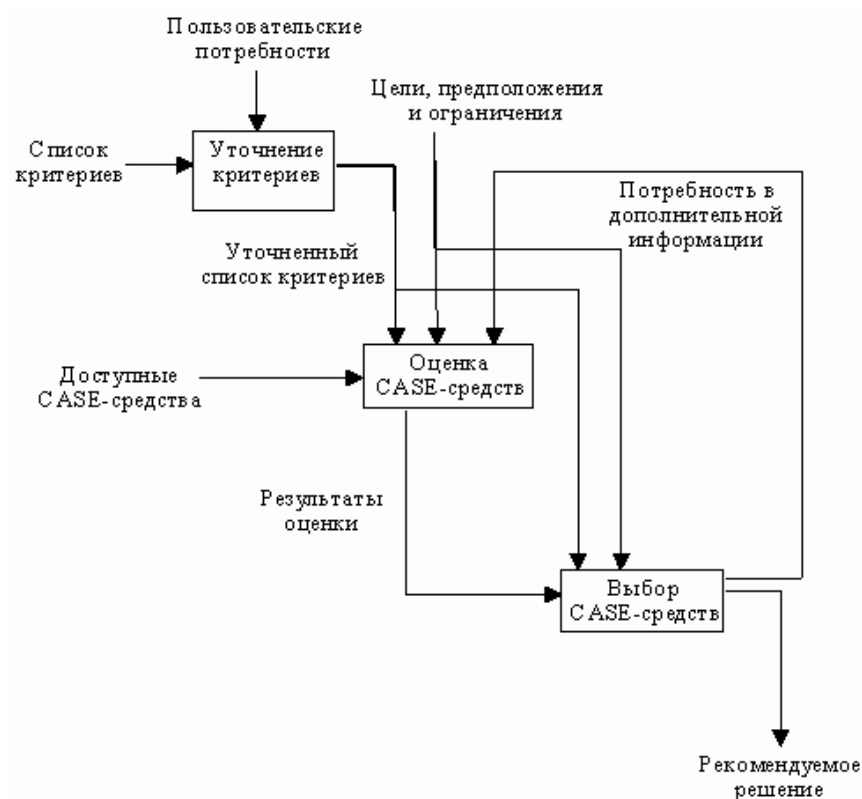


Рис. 1.28

Как видно из рисунка 1.28, входной информацией для процесса оценки является:

- определение пользовательских потребностей;
- цели и ограничения проекта;
- данные о доступных CASE-средствах;
- список критериев, используемых в процессе оценки.

Результаты оценки могут включать результаты предыдущих оценок. При этом не следует забывать, что набор критериев, использовавшихся при предыдущей оценке, должен быть совместимым с текущим набором. Конкретный вариант реализации процесса (оценка и выбор, оценка для будущего выбора или выбор, основанный на предыдущих оценках) определяется перечисленными выше целями.

Элементы процесса включают:

цели, предположения и ограничения, которые могут уточняться в ходе процесса;

потребности пользователей, отражающие количественные и качественные требования пользователей к CASE-средствам;

критерии, определяющие набор параметров, в соответствии с которыми производится оценка и принятие решения о выборе;

формализованные результаты оценок одного или более средств;

рекомендуемое решение (обычно либо решение о выборе, либо дальнейшая оценка).

Процесс оценки и/или выбора может быть начат только тогда, когда лицо, группа или организация полностью определила для себя конкретные потребности и формализовала их в виде количественных и качественных требований в заданной предметной области. Термин “пользовательские требования” далее означает именно такие формализованные требования.

Пользователь должен определить конкретный порядок действий и принятия решений с любыми необходимыми итерациями. Например, процесс может быть представлен в виде дерева решений с его последовательным обходом и выбором подмножеств кандидатов для более детальной оценки. Описание последовательности действий должно определять поток данных между ними.

Определение списка критериев основано на пользовательских требованиях и включает:

выбор критериев для использования из приведенного далее перечня;

определение дополнительных критериев;

определение области использования каждого критерия (оценка, выбор или оба процесса);

определение одной или более метрик для каждого критерия оценки;

назначение веса каждому критерию при выборе.

В заключение приведем примеры комплексов CASE-средств, обеспечивающих поддержку полного ЖЦ ПО. Здесь хотелось бы еще раз отметить нецелесообразность сравнения отдельно взятых CASE-средств, поскольку ни одно из них не решает в целом все проблемы создания и сопровождения ПО. Это подтверждается также полным набором критериев оценки и выбора, которые затрагивают все этапы ЖЦ ПО. Сравниваться могут комплексы методологически и технологически согласованных инструментальных средств, поддерживающие полный ЖЦ ПО и обеспеченные необходимой технической и методической поддержкой со стороны фирм-поставщиков. На сегодняшний день наиболее развитым из всех поставляемых в России комплексов такого рода является комплекс технологий и инструментальных средств создания ИС, основанный на методологии и технологии DATARUN. В состав комплекса входят следующие инструментальные средства:

- CASE-средство Silverrun;
- средство разработки приложений JAM;
- мост Silverrun-RDM <-> JAM;
- комплекс средств тестирования QA;
- менеджер транзакций Tuxedo;
- комплекс средств планирования и управления проектом SE Companion;
- комплекс средств конфигурационного управления PVCS;
- объектно-ориентированное CASE-средство Rational Rose;
- средство документирования SoDA.

Примерами других подобных комплексов являются:

- Vantage Team Builder for Uniface + Uniface (фирмы "DataX/Florin" и "ЛАНИТ");
- комплекс средств, поставляемых и используемых фирмой "ФОРС";
- CASE-средства Designer/2000 (основное), ERwin, Bpwin и Oowin (альтернативные);
- средства разработки приложений Developer/2000, ORACLE Power Objects (основные) и Usoft Developer (альтернативное);
- средство настройки и оптимизации ExplainSQL (Platinum);
- средства администрирования и сопровождения SQLWatch, DBVision, SQL Spy, TSReorg и др. (Platinum);
- средство документирования ORACLE Book;
- комплекс средств на основе продуктов фирмы CENTURA;
- CASE-средства ERwin, Bpwin и Oowin (объектно-ориентированный анализ);
- средства разработки приложений SQLWindows и TeamWindows;
- средство тестирования и оптимизации приложений "клиент-сервер" SQLBench (ARC);
- средства эксплуатации и сопровождения Quest и Crystal Reports.

2. СИСТЕМНОЕ ПО ПЕРЕДАЧИ И ЗАЩИТЫ ДАННЫХ

Децентрализация процессов обработки данных реализовывалась по двум направлениям:

путем подключения к отдельным ЭВМ (или комплексу ЭВМ, объединенных в рамках вычислительного центра (ВЦ)) множества абонентских пунктов пользователей, т.е. создания систем телеобработки данных;

путем создания информационно-вычислительных сетей, в которых осуществлялось объединение между собой множества территориально удаленных друг от друга ЭВМ или ВЦ.

Создание крупных высокоэффективных систем обработки данных связано с объединением средств вычислительной техники, обслуживающей отдельные предприятия, организации и их подразделения, с помощью средств связи в единую распределенную вычислительную систему – **сеть ЭВМ** (совокупность сети передачи данных, взаимосвязанных ею ЭВМ и необходимых для реализации этой связи ПО и технических средств, которая предназначена для организации распределенной обработки информации).

Такой комплекс средств вычислительной техники позволяет повысить эффективность систем обработки информации за счет снижения затрат, повышения надежности и производительности эксплуатируемых ЭВМ, рационального сочетания преимуществ централизованной и децентрализованной обработки информации благодаря приближению средств сбора исходной и выдачи конечной информации непосредственно к местам ее возникновения и потребления, а также комплексного использования единых мощных вычислительных и информационных **ресурсов сети** (программного, технического, информационного и организационного обеспечения информационной сети, предназначенного для решения прикладных задач). Таким образом, приходим к понятию **информационной сети** – совокупности сети ЭВМ и взаимодействующих через нее удаленных реальных оконечных систем, обеспечивающей доступ прикладных процессов, расположенных в любой из этих систем, ко всем ее информационным, вычислительным, коммуникационным ресурсам и коллективное их использование. В дальнейшем будем называть информационную сеть локальной вычислительной сетью (ЛВС).

Передача информации (или **передача данных** – пересылка данных при помощи средств связи из одного места для приёма их в другом) между территориально удаленными компонентами подобных распределенных систем осуществляется в основном с помощью стандартных телефонных и телеграфных каналов, также витых пар проводов и коаксиальных кабелей связи (рис. 2.1).

На рис. 2.1 показаны **функциональные блоки** (ФБ – системы или устройства, выполняющие определенную логически связанную группу функций) аппаратуры передачи и приема, **источник данных**

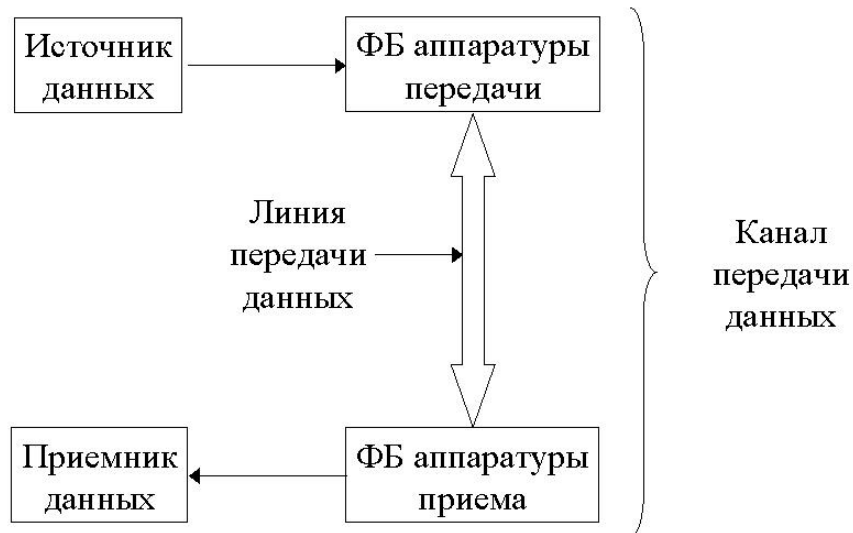


Рис. 2.1

(ФБ, порождающий данные для передачи) и **приемник данных** (ФБ, принимающий переданные данные). Работа ФБ аппаратуры передачи и приема строится в соответствии с **протоколом** передачи данных (набором семантических и синтаксических правил, определяющим поведение ФБ при передаче и приеме данных). После **соединения** (связи, устанавливаемой между ФБ для передачи информации) в соответствии с принятым протоколом осуществляется **обмен данными**. Средства двустороннего обмена данных, представляющие собой совокупность аппаратуры окончания канала данных и линии передачи данных образуют **канал передачи данных**.

Передающей средой называется совокупность линии передачи данных и, возможно, коммутаторов данных, повторителей и другого оборудования, не относящегося к станции данных, организация структуры и функционирование которой обеспечивают физическую передачу данных между станциями.

Современный прогресс в области оптоволоконной техники (использование световодов) позволяет резко повысить пропускную способность линий связи. Так, система F6M обеспечивает передачу информации до 1 Мбит/с, заменяя до 96 телефонных каналов, а система F400M – передачу до 400 Мбит/с информации, заменяя 5760 телефонных каналов.

Расширение состава и совершенствование аппаратуры приема-передачи, а также резкое снижение стоимости вычислительной техники (ВТ) привели к использованию в качестве абонентских пунктов систем

телеобработки данных интеллектуальных терминалов, создаваемых на базе микропроцессоров и микроЭВМ и обеспечивающих частичную обработку информации (главным образом, предварительную обработку исходной информации в виде ее логического контроля, агрегирования и т.д.) непосредственно до ее передачи по каналам связи. Использование интеллектуальных терминалов сближает функциональные возможности систем телеобработки данных и вычислительных сетей. В настоящее время вычислительные сети представляют собой высшую организационную форму применения ЭВМ.

Для современных вычислительных сетей характерно:

объединение многих достаточно удаленных друг от друга ЭВМ и (или) отдельных вычислительных систем в единую распределенную систему обработки данных;

применение средств приема-передачи данных и каналов связи для организации обмена информацией в процессе взаимодействия средств ВТ;

наличие широкого спектра периферийного оборудования, используемого в виде абонентских пунктов и терминалов пользователей, подключаемых к узлам сети передачи данных;

использование унифицированных способов сопряжения технических средств и каналов связи, облегчающих процедуру наращивания и замену оборудования;

наличие операционной системы, обеспечивающей надежное и эффективное применение технических и программных средств в процессе решения задач пользователей вычислительной сети.

Особенностью эксплуатации вычислительных сетей является не только приближение аппаратных средств непосредственно к местам возникновения и использования данных, но и разделение функции обработки и управления на отдельные составляющие с целью их эффективного распределения между несколькими ЭВМ, а также обеспечение надежного и быстрого доступа пользователей к вычислительным и информационным ресурсам и организация коллективного использования этих ресурсов.

Вычислительные сети позволяют автоматизировать управление производством, транспортом, материально-техническим снабжением в масштабе отдельных регионов и страны в целом.

Возможность концентрации в вычислительных сетях больших объемов данных, общедоступность этих данных, а также программных и аппаратных средств обработки и высокая надежность их функционирования – все это позволяет улучшить информационное обслуживание пользователей и резко повысить эффективность применения ВТ.

В условиях вычислительной сети предусмотрена возможность:

организовать параллельную обработку данных многими ЭВМ;

создавать распределенные базы данных, размещаемые в памяти различных ЭВМ;

специализировать отдельные ЭВМ (группы ЭВМ) для эффективного решения определенных классов задач;

автоматизировать обмен информацией и программами между отдельными ЭВМ и пользователями сети;

резервировать вычислительные мощности и средства передачи данных на случай выхода из строя отдельных из них с целью быстрого восстановления нормальной работы сети;

перераспределять вычислительные мощности между пользователями сети в зависимости от изменения их потребностей и сложности решаемых задач;

стабилизировать и повышать уровень загрузки ЭВМ и дорогостоящего периферийного оборудования;

сочетать работу в широком диапазоне режимов: диалоговом, пакетном, режимах «запрос-ответ», а также сбора, передачи и обмена информацией.

Как показывает практика, за счет расширения возможностей обработки данных, лучшей загрузки ресурсов и повышения надежности функционирования системы в целом стоимость обработки данных в вычислительных сетях не менее чем в полтора раза ниже по сравнению с обработкой аналогичных данных на автономных ЭВМ.

2.1. Передача данных

Основная функция систем передачи данных в условиях функционирования вычислительных сетей заключается в организации быстрой и надежной передачи информации произвольным абонентам сети, а также в сокращении затрат на передачу данных. Последнее особенно важно, так как за прошедшее десятилетие произошло увеличение доли затрат на передачу данных в общей структуре затрат на организацию сетевой обработки информации. Это объясняется главным образом тем, что затраты на техническое обеспечение вычислительных сетей сократились за этот период примерно в десять раз, тогда как затраты на организацию и эксплуатацию каналов связи сократились только в два раза.

Важнейшая характеристика сетей передачи данных – время доставки информации – зависит от структуры сети передачи данных, пропускной способности линий связи, а также от способа соединения каналов связи между взаимодействующими абонентами сети и способа передачи данных по этим каналам. В настоящее время различают системы передачи данных с постоянным включением каналов связи {некоммутируемые каналы связи} и коммутацией на время передачи информации по этим каналам.

При использовании *некоммутируемых каналов связи* средства приема-передачи абонентских пунктов и ЭВМ постоянно соединены между собой, т.е. находятся в режиме «on-line». В этом случае отсутствуют потери времени на коммутацию, обеспечиваются высокая степень готовности системы к передаче информации, более высокая надежность каналов связи и, как следствие, достоверность передачи

информации. Недостатками такого способа организации связи являются низкий коэффициент использования аппаратуры передачи данных и линий связи, высокие расходы на эксплуатацию сети. Рентабельность подобных сетей достигается только при условии достаточно полной загрузки этих каналов.

При коммутации абонентских пунктов и ЭВМ только на время передачи информации (т.е. нормальным режимом для которых является режим «off-line») принцип построения узла коммутации определяется способами организации прохождения информации в сетях передачи данных. Существуют три основных способа подготовки и передачи информации в сетях, основанных на коммутации: каналов, сообщений и пакетов (рис. 2.2, здесь пункты А, В и С – пункты коммутации).

2.1.1. Коммутация каналов

Способ коммутации каналов заключается в установлении физического канала связи для передачи данных непосредственно между абонентами сети. При использовании коммутируемых каналов тракт (путь) передачи данных образуется из самих каналов связи и устройств коммутации, расположенных в узлах связи.

Установление соединения заключается в том, что абонент посылает в канал связи заданный набор символов, прохождение которых по сети через соответствующие узлы коммутации вызывает установку нужного соединения с вызываемым абонентом. Этот транзитный канал образуется в начале сеанса связи, остается фиксированным на период передачи всей информации и разрывается только после завершения передачи информации.

Такой способ соединения используется в основном в сетях, где требуется обеспечить непрерывность передачи сообщений (например, при использовании телефонных каналов связи и абонентского телеграфа). В этом случае связь абонентов возможна только при условии использования ими однотипной аппаратуры, одинаковых каналов связи, а также единых кодов.

К достоинствам данного способа организации соединения абонентов сети следует отнести:

- гибкость системы соединения в зависимости от изменения потребностей;

- высокую экономичность использования каналов, достигаемую за счет их эксплуатации только в течение времени установления связи и непосредственно передачи данных;

- невысокие расходы на эксплуатацию каналов связи (на порядок меньше, чем при эксплуатации некоммутируемых линий связи).

Способ коммутации каналов более оперативный, так как позволяет вести непрерывный двусторонний обмен информацией между двумя абонентами.

Недостатками коммутируемых каналов связи является необходи-

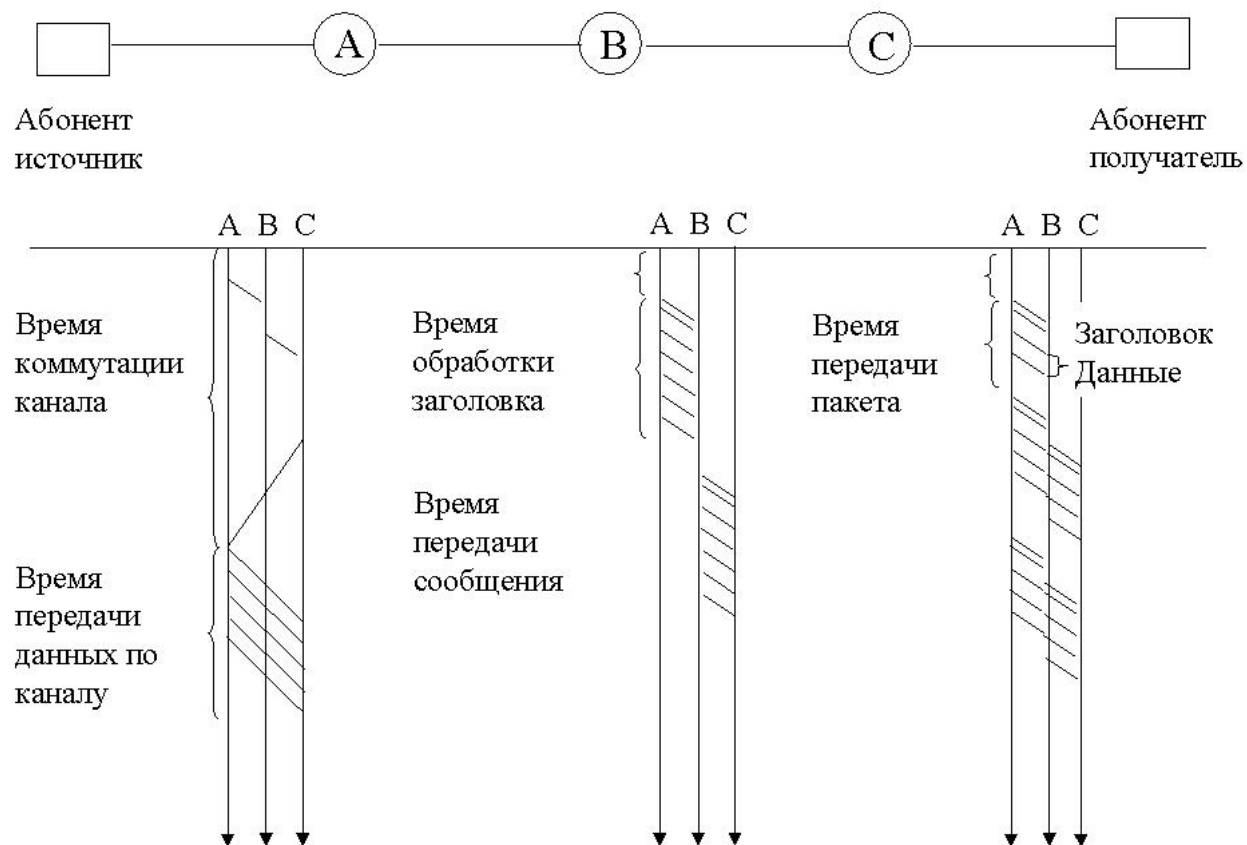


Рис. 2.2

мость использования специальных и коммутирующих устройств, которые снижают скорость передачи данных и достоверность передаваемой информации. Использование специальных методов и средств, обеспечивающих повышение достоверности передачи информации в сети, влечет за собой снижение скорости передачи данных за счет:

увеличения объема передаваемой информации, вызванного необходимостью введения избыточных знаков;

потерь времени на кодирование информации в узле-передатчике и декодирование, логический контроль и другие преобразования – в узле-приемнике.

Наконец, сокращение потоков информации ниже пропускной способности аппаратной части и каналов связи ведет к недогрузке вала, а в период пиковой нагрузки может вызвать определенные потери вызовов.

2.1.2. Коммутация сообщений

При коммутации сообщений поступающая на узел связи информация передается в память узла связи, после чего анализируется адрес получателя. В зависимости от занятости требуемого канала сообщение либо передается в память соседнего узла, либо становится в очередь для последующей передачи. Таким образом, способ коммутации сообщений обеспечивает поэтапный характер передачи информации. В этом случае сообщения содержат адресный признак (заголовки), в соответствии с которым осуществляется автоматическая передача информации в сети от абонента-передатчика к абоненту-приемнику. Все функции согласования работы отдельных участков сети связи, а также управление передачей сообщений и их соответствующую обработку выполняют центры (узлы) коммутации сообщений. Основное функциональное назначение центра коммутации сообщений – обеспечить автоматическую передачу информации от абонента к абоненту в соответствии с адресным признаком сообщения и требованиями к качеству и надежности связи.

Метод коммутации сообщений обеспечивает независимость работы отдельных участков сети, что значительно повышает эффективность использования каналов связи при передаче одного и того же объема информации (которая в этом случае может достигать 80-90 % от максимального значения). В системе с коммутацией сообщений происходит сглаживание несогласованности в пропускной способности каналов и более эффективно реализуется передача многоадресных сообщений (так как не требуется одновременного освобождения всех каналов между узлом-передатчиком и узлом-приемником). Передача информации может производиться в любое время, так как прямая связь абонентов друг с другом необязательна. К недостаткам метода следует отнести односторонний характер связи между абонентами сети. Для

более полной загрузки каналов и их эффективного использования возможно совместное применение перечисленных методов коммутации, основой которого служат следующие условия:

использование в одном и том же узле связи аппаратуры для коммутации каналов и для коммутации сообщений (выбор того или иного способа коммутации в узле осуществляется в зависимости от загрузки каналов связи);

организация сети с коммутацией каналов для узлов верхних уровней иерархии и коммутации сообщений для нижних уровней.

2.1.3. Коммутация пакетов

В последние годы появился еще один способ коммутации абонентов сети – так называемая **коммутация пакетов**. Этот способ сочетает в себе ряд преимуществ методов коммутации каналов и коммутации сообщений. При коммутации пакетов перед началом передачи сообщение разбивается на короткие пакеты фиксированной длины, которые затем передаются по сети. В пункте назначения эти пакеты вновь объединяются в первоначальное сообщение, а так как их длительное хранение в запоминающем устройстве узла связи не предполагается, пакеты передаются от узла к узлу с минимальной задержкой во времени. В этом отношении указанный метод близок методу коммутации каналов.

При коммутации пакетов их фиксированная длина обеспечивает эффективность обработки пакетов, предотвращает блокировку линий связи и значительно уменьшает емкость требуемой промежуточной памяти узлов связи. Кроме того, сокращается время задержки при передаче информации, т.е. скорость передачи информации превышает аналогичную скорость при методе коммутации сообщений.

К недостаткам метода следует отнести односторонний характер связи между абонентами сети.

Различают два основных типа систем связи с коммутацией пакетов:

- 1) в системах первого типа устройство коммутации анализирует адрес места назначения каждого принятого пакета и определяет канал, необходимый для передачи информации;
- 2) в системах второго типа пакеты рассылаются по всем каналам и терминалам, каждый канал (терминал), в свою очередь, проанализировав адрес места назначения пакета и сравнив его с собственным, осуществляет прием и дальнейшую передачу (обработку) пакета либо игнорирует его.

Первый тип систем коммутации пакетов характерен для глобальных сетей с огромным числом каналов связи и терминалов, второй тип применим для сравнительно замкнутых сетей с небольшим числом абонентов.

2.1.4. Сопряжение ЭВМ и устройств в сетях

Существенное влияние на организацию систем обработки данных оказывают технические возможности средств, используемых для сопряжения (комплектования) ЭВМ и других устройств. Основным элементом сопряжения является интерфейс, определяющий число линий, используемых для передачи сигналов и данных, а также способ (алгоритм) передачи информации по линиям связи.

Все интерфейсы, используемые в ВТ и сетях, разделяются на три вида: *параллельные, последовательные, связные*.

Параллельный интерфейс состоит из большого числа линий, по которым передача данных осуществляется в параллельном коде (обычно в виде 8-128 разрядных слов). Параллельный интерфейс обладает большой пропускной способностью: порядка 10^4 - 10^5 бод (бит/с). Столь большие скорости передачи данных обеспечиваются за счет ограниченной длины интерфейса (обычно от нескольких метров до десятков (очень редко до сотни) метров).

Последовательный интерфейс состоит, как правило, из одной линии, данные по которой передаются в последовательном коде. Пропускная способность последовательного интерфейса составляет 10^3 - 10^4 бит/с при длине линии интерфейса от десятков метров до километра.

Связные интерфейсы содержат каналы связи, работа которых обеспечивается аппаратурой передачи данных (АПД), повышающей (в основном с помощью специальных физических методов) достоверность передачи данных. Связные интерфейсы обеспечивают передачу данных на любые расстояния, однако, с небольшой скоростью (в пределах 10^2 - 10^3 бит/с). Применение связных интерфейсов экономически целесообразно на расстоянии не менее километра.

В многопроцессорных и многомашинных вычислительных системах используются в основном параллельные интерфейсы для сопряжения отдельных устройств в ЭВМ и только в отдельных случаях – последовательные интерфейсы для подключения периферийных устройств. В распределенных системах из-за значительных расстояний между устройствами применяются последовательные и связные интерфейсы. От организации интерфейсов между устройствами во многом зависит организация программного обеспечения.

2.2. Информационные сети

Прогресс в развитии микропроцессорной техники сделал ее доступной массовому потребителю, а высокая надежность, относительно низкая стоимость, простота общения с пользователем-непрофессионалом в области вычислительной техники послужили основой для организации систем распределенной обработки данных, включающих от десятка до сотен ПЭВМ, объединенных в вычислительные сети. В отличие от вычислительных сетей, создаваемых на базе больших ЭВМ

и охватывающих значительную территорию, сети на базе ПЭВМ получили название локальных, так как они ориентированы в первую очередь на объединение вычислительных машин и периферийных устройств, сосредоточенных на небольшом пространстве (например, в пределах одного помещения, здания, группы зданий в пределах нескольких километров). Появление ЛВС позволило значительно повысить эффективность применения ВТ за счет более рационального использования аппаратных, программных и информационных ресурсов вычислительной системы, значительного улучшения эксплуатационных характеристик (в первую очередь повышения надежности) и создания максимальных удобств для работы конечных пользователей. Сравнительно низкая стоимость, высокая живучесть и простота комплектования и эксплуатации ЛВС, оснащенные современными операционными системами различного назначения, высокоскоростными средствами передачи данных, оперативной и внешней памятью большой емкости способствовали их быстрому распространению для автоматизации управленческой деятельности в учреждениях, на предприятиях, а также для создания на их основе информационных, измерительных и управляющих систем автоматизации технологических и производственных процессов. Одной из главных проблем создания локальных вычислительных сетей является проблема аппаратной совместимости ВТ. В настоящее время вычислительные средства ЛВС в основном объединяются с помощью высокоскоростных либо низкоскоростных каналов передачи данных. Такие вычислительные сети получили название свободносвязанных, так как протекание вычислительных процессов в них может осуществляться асинхронно. При незначительной удаленности вычислительного оборудования более эффективным средством связи между отдельными аппаратными компонентами ЛВС является последовательный интерфейс. Его достаточно высокая пропускная способность позволяет иметь единственный канал передачи данных – моноканал; при этом работа всей системы осуществляется в режиме мультиплексирования.

Для организации связи в ЛВС используются два метода коммутации: с частотным и временным разделением каналов. При этом элементами коммутации служат каналы и пакеты (см. рис. 2.2).

При коммутации каналов выделяется единственный канал (с частотным или временным разделением) на весь сеанс связи. При коммутации пакетов канал связи выделяется только на время, необходимое для передачи одного пакета.

2.2.1. Классификация ЛВС

Все множество видов ЛВС можно разделить на четыре группы.

К *первой группе* относятся ЛВС, ориентированные на массового пользователя. Такие ЛВС объединяют в основном персональные ЭВМ с помощью систем передачи данных, имеющих низкую стоимость и обеспечивающих передачу информации на расстояние 100-500 м со

скоростью 2400-19200 бод.

Ко *второй группе* относятся ЛВС, объединяющие, кроме ПЭВМ, микропроцессорную технику, встроенную в технологическое оборудование (средства автоматизации проектирования, обработки документальной информации, кассовые аппараты и т.д.), а также средства электронной почты. Система передачи данных таких ЛВС обеспечивает передачу информации на расстояние до 1 км со скоростью от 19 200 бод до 1 Мбод. Стоимость передачи данных в таких сетях примерно на 30% превышает стоимость этих работ в сетях первой группы.

К *третьей группе* относятся ЛВС, объединяющие ПЭВМ, мини-ЭВМ и ЭВМ среднего класса. Эти ЛВС используются для организации управления сложными производственными процессами с применением робототехнических комплексов и гибких автоматизированных модулей, а также для создания крупных систем автоматизации проектирования, систем управления научными исследованиями и т.п. Системы передачи данных в таких ЛВС имеют среднюю стоимость и обеспечивают передачу информации на расстояние до нескольких километров со скоростью 120 Мбод.

Для ЛВС *четвертой группы* характерно объединение в своем составе всех классов ЭВМ. Такие ЛВС применяются в сложных системах управления крупным производством и даже отдельной отраслью: они включают в себя основные элементы всех предыдущих групп ЛВС. В рамках данной группы ЛВС могут применяться различные системы передачи данных, в том числе обеспечивающие передачу информации со скоростью от 10 до 50 Мбод на расстояние до 10 км. По своим функциональным возможностям ЛВС этой группы мало чем отличаются от региональных вычислительных сетей, обслуживающих крупные города, районы, области. В своем составе они могут содержать разветвленную сеть соединений между различными абонентами-отправителями и получателями информации.

По топологическим признакам ЛВС делятся на сети следующих типов: *с общей шиной, кольцевые, иерархические, радиальные и многосвязные*.

В ЛВС *с общей шиной* (рис. 2.3, а) одна из машин, как правило, служит в качестве системного обслуживающего устройства, обеспечивающего централизованный доступ к общим файлам и базам данных, печатающим устройствам и другим вычислительным ресурсам. ЛВС данного типа приобрели большую популярность благодаря низкой стоимости, высокой гибкости и скорости передачи данных, легкости расширения сети (подключение новых абонентов к сети не сказывается на ее основных характеристиках). К недостаткам шинной топологии следует отнести необходимость использования довольно сложных протоколов и уязвимость в отношении физических повреждений кабеля.

Кольцевая топология (рис. 2.3, б) характеризуется тем, что информация по кольцу может передаваться только в одном направлении и все подключенные ПЭВМ могут участвовать в ее приеме и передаче.

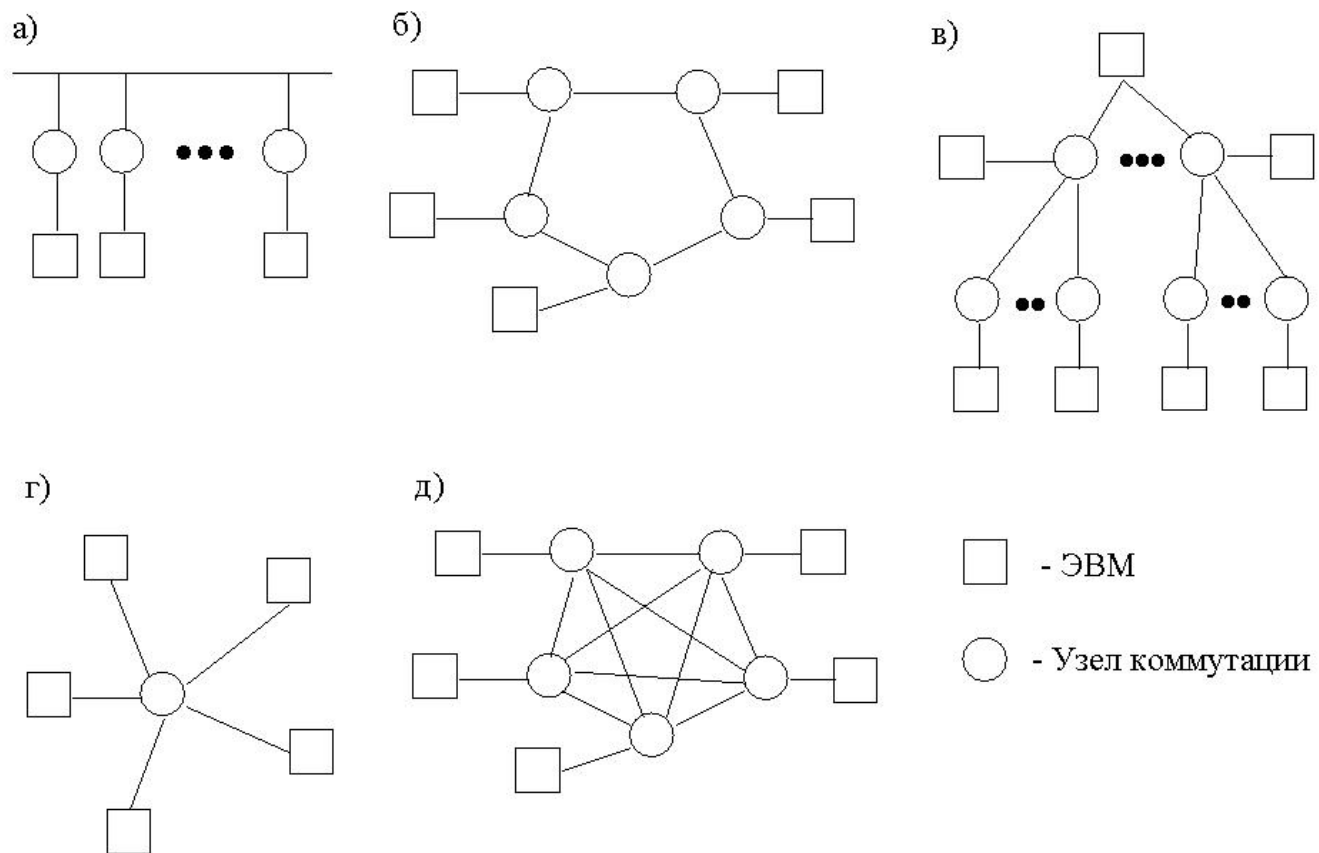


Рис. 2.3

При этом абонент-получатель должен пометить полученную информацию специальным маркером, иначе могут появиться «заблудившиеся» данные, мешающие нормальной работе сети. Как последовательная конфигурация кольцо особенно уязвимо в отношении отказов: выход из строя какого-либо сегмента кабеля приводит к прекращению обслуживания всех пользователей. Разработчики ЛВС приложили немало усилий, чтобы справиться с этой проблемой. Защита от повреждений или отказов обеспечивается либо замыканием кольца на обратный (дублирующий) путь, либо переключением на запасное кольцо. И в том, и в другом случае сохраняется общая кольцевая топология.

Иерархическая ЛВС (конфигурация типа «дерево») представляет собой более развитый вариант структуры ЛВС, построенной на основе общей шины (см. рис. 2.3, в). Дерево образуется путем соединения нескольких шин с корневой системой, где размещаются самые важные компоненты ЛВС. Оно обладает необходимой гибкостью для того, чтобы охватить средствами ЛВС несколько этажей в здании или несколько зданий на одной территории, и реализуется, как правило, в сложных системах, насчитывающих десятки и даже сотни абонентов.

Радиальную (звездообразную) конфигурацию (см. рис. 2.3, г) можно рассматривать как дальнейшее развитие структуры «дерево с корнем» с ответвлением к каждому подключенному устройству. В центре сети обычно размещается коммутирующее устройство, обеспечивающее жизнеспособность системы. ЛВС подобной конфигурации находят наиболее частое применение в автоматизированных учрежденческих системах управления, использующих центральную базу данных. Звездообразные ЛВС, как правило, менее надежны, чем с общей шиной или иерархические, но эта проблема решается дублированием аппаратуры центрального узла. К недостаткам можно также отнести значительное потребление кабеля (иногда в несколько раз превышающее расход в аналогичных по возможностям ЛВС с общей шиной или иерархических).

Наиболее сложной и дорогой является *многосвязная топология*, в которой каждый узел связан со всеми другими узлами сети (см. рис. 2.3, д). Эта топология в ЛВС применяется очень редко, в основном там, где требуются исключительно высокие надежность сети и скорость передачи данных.

На практике чаще встречаются гибридные ЛВС, приспособленные к требованиям конкретного заказчика и сочетающие фрагменты шинной, звездообразной и других топологий. Важное значение имеет выбор правильной **архитектуры сети** (логической структуры и принципов работы информационной сети).

2.3. Взаимосвязь открытых систем

Характер топологии сети оказывает влияние на организацию обмена информацией между ее абонентами. Как правило, такой обмен информацией между абонентами сети осуществляется с помощью

фиксированных блоков (фрагментов) информации, которые называют **пакетами**. Любой пакет, независимо от типа структуры ЛВС, включает в себя адреса получателя и отправителя, собственно данные и, как правило, контрольную сумму (рис. 2.4).

Адрес получателя	Адрес отправителя	Данные	Контрольная сумма
---------------------	----------------------	--------	----------------------

Рис. 2.4

Каждое устройство принимает пакеты, которые ему адресованы, проверяет корректность полученных данных по контрольной сумме и посылает соответствующий ответ устройству-отправителю. Поскольку одно устройство может получать пакеты от нескольких устройств, адрес отправителя является обязательной частью формата.

Стремительное развитие ЛВС, использование в них технических и программных средств, производимых часто в разных странах, сделали необходимым принятие международных соглашений, стандартов на передачу информации, т.е. на характеристики каналов связи, стыковки ПЭВМ с каналами связи, правила стыковки сетей друг с другом, протоколы межмашинного и межсетевого взаимодействия и т.д. Концепции сетевых протоколов развивались в последние 25 лет и были направлены на решение следующих задач:

1. Обеспечить логическую декомпозицию сложной сети на меньшие, более понятные части (уровни).
2. Обеспечить стандартные интерфейсы между сетевыми функциями, например, стандартные интерфейсы между модулями ПО.
3. Обеспечить симметрию в отношении функций, реализуемых в каждом узле сети. Каждый уровень в некотором узле сети выполняет те же функции, какие выполняются аналогичным уровнем в другом узле.
4. Обеспечить средства предсказания изменений и управления изменениями, которые могут быть внесены в сетевую логику (ПО или микропрограммы).
5. Обеспечить простой стандартный язык коммуникации разработчиков сетей, администраторов, фирм-поставщиков и пользователей, используемый при обсуждении сетевых функций.

2.3.1. Уровни взаимодействия в ЛВС

Международная организация по стандартизации ISO (International Standard Organization) подготовила проект эталонной модели взаимодействия открытых информационных сетей (ВОС) (**открытая информационная сеть** – информационная сеть, в которой взаимодействие

всех входящих в ее состав реальных систем, а также ее взаимодействие с реальными системами, не входящими в ее состав, подчиняется требованиям Международной организации по стандартизации). Модель разработана и принята в качестве международного стандарта и включает семь уровней, характеризующих любую существующую систему связи и взаимодействующих на строго иерархической основе по принципу «снизу вверх». Определены следующие уровни взаимодействия: *физический, канальный, сетевой, транспортный, сеансовый, уровень представления данных и прикладной уровень* (рис. 2.5). На основе понятия уровней введено также понятие **службы взаимосвязи открытых систем** – совокупность технических и (или) программных средств, входящих в состав открытой информационной системы, реализующая согласованные по горизонтали и вертикали функции некоторого уровня и всех расположенных ниже уровней базовой эталонной модели ВОС и выполняющая функции поставщика сервиса для компонентов сети, которые расположены над ее границей.

Физический и канальный уровни образуют нижнюю группу и непосредственно связаны с каналом передачи данных.

Физический уровень осуществляет сопряжение с каналом. Функции на этом уровне обеспечивают активизацию, поддержку и деактивацию

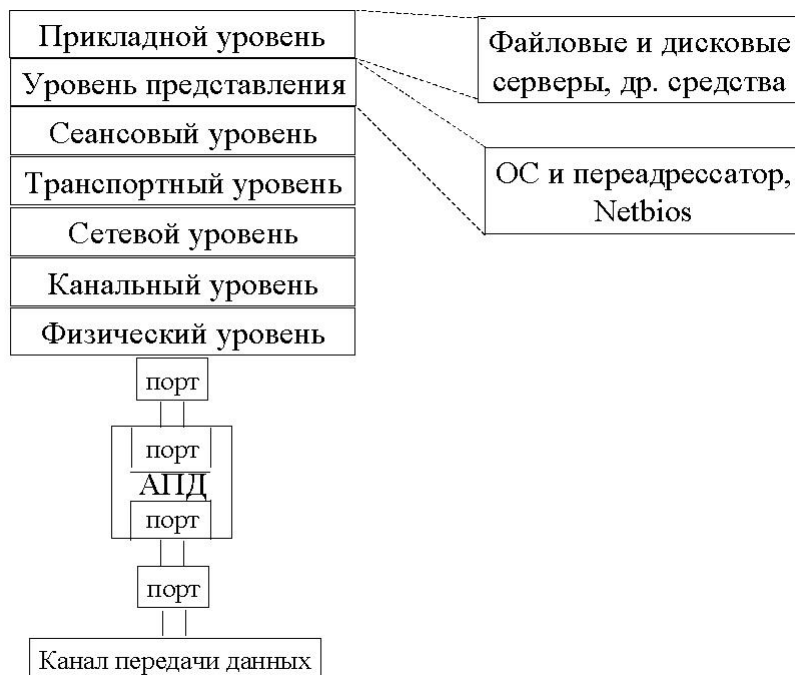


Рис. 2.5

физической цепи между конечным оборудованием данных (ООД) и АПД.

Канальный уровень отвечает за передачу данных по каналу. Он обеспечивает синхронизацию данных для разграничения потока битов из физического уровня. Он обеспечивает также вид представления битов, создает определенные гарантии для того, чтобы данные благополучно прибывали в принимающее ООД. Он обеспечивает управление потоком данных, чтобы гарантировать, что ООД не будет перегружено в любой момент времени слишком большим количеством данных. Одна из его наиболее важных функций состоит в обнаружении ошибок передачи и обеспечении механизма восстановления в случае потери данных, их дублирования или ошибок данных.

В следующую группу входят *сетевой и транспортный уровни*, которые «прокладывают» путь информации между системой-отправителем и системой-получателем и управляют процессом передачи по этому пути.

Сетевой уровень определяет интерфейс ООД пользователя с сетью пакетной коммутации, а также интерфейс двух устройств ООД друг с другом в сети пакетной коммутации. Он также определяет маршрутизацию в сети и связь между сетями (интерсетевой протокол).

Транспортный уровень обеспечивает интерфейс между сетью передачи данных и верхними тремя уровнями (как правило, находящимися в месторасположении пользователя). Именно этот уровень предоставляет пользователю факультативные возможности получения сервиса определенного качества (и стоимости) от самой сети (т.е. сетевого уровня).

Третью группу образуют *сеансовый, уровень представления данных и прикладной уровни*. Они непосредственно связаны с организацией взаимодействия прикладных программ пользователей, а также с вводом, хранением, обработкой данных и выдачей результатов. Все процессы, проходящие на перечисленных уровнях, носят название прикладных. Это главные процессы в коммутационных системах; именно ради них создаются сети, в том числе ЛВС.

Сеансовый уровень служит интерфейсом пользователя с уровнем транспортных услуг. Этот уровень обеспечивает средства организации обмена данными между пользователями, и пользователи могут выбрать типы синхронизации и управления, которые требуются от уровня, такие как:

1. Поочередно двунаправленный диалог или одновременно двунаправленный диалог.
2. Точки синхронизации для промежуточного контроля и восстановления при передаче файлов.
3. Аварийное окончание и рестарты.
4. Нормальная и ускоренная передача данных.

Уровень представления данных определяет синтаксис данных в модели, т.е. представление данных. Его главная роль, например, состоит

в том, чтобы принимать типы данных (знак, целое число) из прикладного уровня и затем согласовывать с уровнем того же ранга синтаксическое представление (такое, как ASCII). Уровень представления обеспечивает отображение данных на виртуальном терминале, а также обеспечивает такие услуги, как разрешение приема электронного сообщения от уровня приложений и согласование с одноранговым уровнем вида представления страницы (например, для типографского набора) для другого прикладного уровня.

Прикладной уровень занимается поддержкой прикладного процесса конечного пользователя. В отличие от уровня представления данных этот уровень имеет дело с семантикой данных. Уровень содержит сервисные элементы для поддержки прикладных процессов, таких как управление трудовыми ресурсами, обмен финансовыми данными, передача/прием языка программирования и обмен деловыми данными. Этот уровень также поддерживает концепции виртуального терминала и виртуального файла.

Каждый из названных выше уровней выполняет указания уровня, расположенного над ним. Так, физический уровень обслуживает канальный уровень, который принимает распоряжения сетевого уровня и т.д. В результате прикладной уровень использует сервис всех остальных уровней процессов взаимодействия.

Задача всех семи уровней – обеспечить надежное взаимодействие прикладных (информационных) процессов. При этом каждый уровень выполняет возложенную на него задачу. Однако уровни работают так, чтобы в нужных случаях можно было проверить работу других уровней. Например, если канальный уровень случайно пропустит ошибку, появившуюся при передаче информации, то ее определит и исправит транспортный уровень.

2.4. Вирусы и антивирусное ПО

2.4.1. Общие сведения о вирусах

Компьютерным вирусом называется программа (некоторая совокупность выполняемого кода/инструкций), которая способна создавать свои копии (не обязательно полностью совпадающие с оригиналом) и внедрять их в различные объекты/ресурсы компьютерных систем, сетей и т.д. без ведома пользователя. При этом копии сохраняют способность дальнейшего распространения. Как и любая обычная программа, «агрессивная» программа должна получить доступ к процессору, т.е. должна быть запущена. Эти программы могут быть резидентными и нерезидентными, но независимо от размещения являются потенциально опасными.

Хотя вирусные атаки случаются не очень часто, общее число вирусов слишком велико, а ущерб от «хулиганских» действий вируса в системе может оказаться значительным. Существуют вирусы, которые

могут привести к потере программ, уничтожить данные, стереть необходимую для работы компьютера информацию, записанную в системных областях памяти, привести к серьезным сбоям в работе компьютера. В результате этих действий вы можете навсегда потерять данные, необходимые для работы, и понести существенный моральный и материальный ущерб. “Эпидемия” компьютерного вируса в фирме может полностью дестабилизировать ее работу. При этом может произойти сбой в работе как отдельных компьютеров, так и компьютерной сети в целом, что повлечет за собой потерю информации, необходимой для нормальной работы, и потерю времени, которое будет затрачено на восстановление данных и приведение компьютеров и/или сети в рабочее состояние.

Внешне действие вирусов может выражаться в том, что периодически, например, по достижении определенного времени, вирус активизируется и выполняет какие-либо операции. В частности вирусы могут: отображать на экране посторонние надписи и символы, “осыпать” символы, уже отображенные на экране, перезагружать компьютер, замедлять работу компьютера, исполнять всевозможные мелодии, удалять файлы и каталоги, стирать выбранные случайным образом сектора жестких и гибких дисков.

Принципиально вирусы могут выполнять любые действия. Они ограничиваются только фантазией автора вируса и возможностями компьютера.

Троянские программы встречаются значительно реже, чем вирусы. Троянские программы представляют собой программы, выполняющие какие-либо полезные функции, однако в определенный момент они могут дополнительно производить некоторые зловредные действия, например, форматировать жесткий диск.

В настоящее время распространение вирусов достигло поистине гигантских размеров и грозит пользователям безвозвратной потерей данных, хранимых на компьютере.

Вирусы можно разделить на классы по следующим признакам:

- по среде обитания вируса;
- по способу заражения среды обитания;
- по деструктивным возможностям;
- по особенностям алгоритма вируса.

По среде обитания вирусы можно разделить на сетевые, файловые и загрузочные. Сетевые вирусы распространяются по компьютерной сети, файловые внедряются в выполняемые файлы, загрузочные – в загрузочный сектор диска (Boot-сектор) или в сектор, содержащий системный загрузчик винчестера (Master Boot Record). Существуют сочетания – например, файлово-загрузочные вирусы, заражающие как файлы, так и загрузочные сектора диска. Такие вирусы, как правило, имеют довольно сложный алгоритм и часто применяют оригинальные методы проникновения в систему.

Способы заражения делятся на резидентный и нерезидентный.

Резидентный вирус при инфицировании компьютера оставляет в оперативной памяти свою резидентную часть, которая затем перехватывает обращения операционной системы к объектам заражения 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. Некоторые вирусы оставляют в памяти компьютера небольшие резидентные программы, которые не распространяют вирус. Такие вирусы считаются нерезидентными.

По деструктивным возможностям вирусы можно разделить на:

- безвредные, т.е. никак не влияющие на работу компьютера (кроме уменьшения свободной памяти на диске в результате своего распространения);
- неопасные, влияние которых ограничивается уменьшением свободной памяти на диске и графическими, звуковыми и прочими эффектами;
- опасные вирусы, которые могут привести к серьезным сбоям в работе;
- очень опасные, которые могут привести к потере программ, уничтожить данные, стереть необходимую для работы компьютера информацию, записанную в системных областях памяти.

По особенностям алгоритма можно выделить следующие группы вирусов:

- 1) *компаньон-вирусы* (companion) – это вирусы, не изменяющие файлы. Алгоритм работы этих вирусов состоит в том, что они создают для EXE-файлов файлы-спутники, имеющие то же самое имя, но с расширением .COM, например, для файла XCOPY.EXE создается файл XCOPY.COM. Вирус записывается в COM-файл и никак не изменяет EXE-файл. При запуске такого файла DOS первым обнаружит и выполнит COM-файл, т.е. вирус, который затем запустит и EXE-файл;
- 2) *вирусы-“черви”* (worm) – вирусы, которые распространяются в компьютерной сети и, так же, как и компаньон-вирусы, не изменяют файлы или сектора на дисках. Они проникают в память компьютера из компьютерной сети, вычисляют сетевые адреса других компьютеров и рассылают по этим адресам свои копии. Такие вирусы иногда создают рабочие файлы на дисках системы, но могут вообще не обращаться к ресурсам компьютера (за исключением оперативной памяти). К счастью, в вычислительных сетях IBM-компьютеров такие вирусы пока не завелись;
- 3) *“паразитические”* – все вирусы, которые при распространении своих копий обязательно изменяют содержимое дисковых секторов или файлов. В эту группу относятся все вирусы, которые не являются “червями” или “компаньон”;
- 4) *“студенческие”* – крайне примитивные вирусы, часто нерезидентные и содержащие большое число ошибок;

- 5) “стелс”-вирусы (вирусы-невидимки, stealth), представляющие собой весьма совершенные программы, которые перехватывают обращения DOS к пораженным файлам или секторам дисков и “подставляют” вместо себя незараженные участки информации. Кроме этого, такие вирусы при обращении к файлам используют достаточно оригинальные алгоритмы, позволяющие “обманывать” резидентные антивирусные мониторы;
- 6) “полиморфик”-вирусы (самошифрующиеся или вирусы-призраки, polymorphic) – достаточно трудно обнаруживаемые вирусы, не имеющие сигнатур, т.е. не содержащие ни одного постоянного участка кода. В большинстве случаев два образца одного и того же полиморфик-вируса не будут иметь ни одного совпадения. Это достигается шифрованием основного тела вируса и модификациями программы-расшифровщика;
- 7) *макровирусы* – вирусы этого семейства используют возможности макроязыков, встроенных в системы обработки данных (текстовые редакторы, электронные таблицы и т.д.). В настоящее время наиболее распространены макровирусы, заражающие текстовые документы редактора Microsoft Word.

Основные симптомы вирусного поражения следующие:

- замедление работы некоторых программ;
- увеличение размеров файлов (особенно выполняемых);
- появление не существовавших ранее “странных” файлов;
- уменьшение объема доступной оперативной памяти (по сравнению с обычным режимом работы);
- внезапно возникающие разнообразные видео- и звуковые эффекты.

При всех перечисленных выше симптомах, а также при других “странных” проявлениях в работе системы (неустойчивая работа, частые “самостоятельные” перезагрузки и прочее) рекомендуется немедленно произвести проверку системы на наличие вирусов с помощью какой-либо антивирусной программы. При этом лучше, если программа будет иметь самую последнюю версию и самые свежие обновления антивирусных баз.

Одним из основных методов борьбы с вирусами является, как и в медицине, своевременная профилактика. Компьютерная профилактика состоит из небольшого количества правил, соблюдение которых значительно снижает вероятность заражения вирусом и утери каких-либо данных. К ним относятся следующие меры.

- Обязательно делайте регулярное резервное копирование.
- Покупайте дистрибутивные копии программного обеспечения у официальных продавцов.
- Создайте системную дискету. Запишите на нее антивирусные программы. Защитите дискету от записи.
- Периодически сохраняйте файлы, с которыми ведется работа, на внешний носитель, например, на дискеты.

- Проверяйте перед использованием все дискеты. Не запускайте непроверенные файлы, в том числе полученные по компьютерным сетям.
- Ограничьте круг лиц, допущенных к работе на конкретном компьютере.
- Периодически проверяйте компьютер на наличие вирусов. При этом пользуйтесь свежими версиями антивирусных программ.

2.4.2. Механизм действия вирусов

Кратко охарактеризуем вирусы и их влияние на работу компьютера.

ФАЙЛОВЫЕ ВИРУСЫ. Областью обитания файловых вирусов являются файлы. Вирусы записывают свой код в тело программного файла таким образом, что при запуске программы вирус первым получает управление. Сделав свое черное дело, вирус передает управление зараженной программе, так что пользователь ничего не замечает. При запуске вирус сканирует локальные диски компьютера и сетевые каталоги в поисках очередной жертвы. После того как подходящий программный файл будет найден, вирус записывает в него свой код.

Механизм распространения файловых вирусов достаточно прост. Создатель вируса намеренно заражает какой-либо программный файл и записывает его на электронную доску объявлений BBS, FTP-сервер, посылает в телеконференцию или отдает приятелю. Как правило, для заражения выбирается что-нибудь интересное: новая игра, самораскрывающийся архив с привлекательным названием или новая версия популярной программы.

Получив новую игру от хорошего знакомого, даже бывалые системные администраторы и опытные программисты не всегда могут удержаться от того, чтобы сразу же ее запустить. Результат может оказаться плачевным. Следовало бы вначале проверить программу антивирусами, но такая проверка тоже не дает полной гарантии – каждый день появляются все новые и новые вирусы.

Самый простой способ гарантированно удалить вирусы с компьютера заключается в том, чтобы после форматирования диска компьютера на низком уровне установить операционную систему и прикладные программы с лицензионных дистрибутивов, и в дальнейшем избегать использования нелегальных, бесплатных (Freeware) и условно-бесплатных (Shareware) программ.

Однако в жизни так бывает очень редко. Даже из числа тех, кто пользуется только лицензионными программами, найдется немало людей, у кого есть условно-бесплатные архиваторы, демонстрационные версии игр, бесплатные средства доступа к Internet или аналогичные средства. Свободный обмен этими программами может привести к вирусному заражению.

Но и в том случае, если вы не пользуетесь программами

сомнительного происхождения, вы можете получить относительно новую разновидность файлового вируса – макрокомандный вирус, распространяющийся с документами офисных приложений, таких как Microsoft Word for Windows или Microsoft Excel for Windows. Документы офисных приложений содержат в себе не только текст и графические изображения, но и макрокоманды, которые представляют собой ничто иное как программы. Эти программы составляются на языке, напоминающем Бейсик. Вирус может изменять существующие макрокоманды и добавлять новые, внедряя свое тело в файл документа.

Механизм распространения макрокомандных вирусов основан на том, что существуют макрокоманды, которые запускаются при открывании документа для редактирования или при выполнении других операций. Разработчик макрокомандного вируса берет безобидный файл с именем, например readme.doc, и записывает в него одну или несколько вирусных макрокоманд, например, вирусную макрокоманду с именем AutoExec. Когда вы открываете такой файл при помощи текстового процессора Microsoft Word for Windows, эта макрокоманда будет автоматически запущена на выполнение. При этом вирус получит управление и может заразить другие документы, хранящиеся на ваших дисках. Если вирусная макрокоманда имеет имя FileSaveAs, то распространение вируса будет происходить при сохранении документа.

Для предотвращения заражения макрокомандными вирусами вы должны перед просмотром или редактированием проверять новые файлы документов с помощью антивирусных программ, способных искать такие вирусы.

ЗАГРУЗОЧНЫЕ ВИРУСЫ. Вторая большая группа вирусов – это так называемые загрузочные вирусы. Распространение и активизация этих вирусов происходит в момент загрузки операционной системы, еще до того как пользователь успел запустить какую-либо антивирусную программу.

Для того чтобы вам был понятнее механизм распространения загрузочных вирусов, напомним, как протекает процесс начальной инициализации компьютера и загрузки операционной системы.

Сразу после включения электропитания компьютера начинает работать программа инициализации, записанная в постоянном запоминающем устройстве базовой системы ввода/вывода BIOS. Эта программа проверяет оперативную память и другие устройства компьютера, а затем передает управление программе начальной загрузки, которая также находится в BIOS.

Программа начальной загрузки пытается записать в оперативную память содержимое самого первого сектора нулевой дорожки жесткого диска, в котором находится главная загрузочная запись Master Boot Record (MBR), либо содержимое самого первого сектора нулевой дорожки дискеты, вставленной в устройство A:. Этот сектор содержит загрузочную запись Boot Record (BR).

Выбор способов начальной загрузки зависит от многих факторов. Если компьютер не оборудован жестким диском, то программа начальной загрузки попытается прочитать загрузочную запись с дискеты. Но такая попытка будет предпринята только в том случае, если в таблице конфигурации компьютера указано, что накопитель на флоппи-диске присутствует. Напомним, что таблица конфигурации компьютера хранится в энергонезависимой памяти и может изменяться при помощи программы BIOS Setup.

Если в компьютере имеется жесткий диск и он правильно описан в таблице конфигурации компьютера, последовательность загрузки зависит от выбора, сделанного при помощи все той же программы BIOS Setup. Пользователь может указать, что компьютер должен загружаться либо только с жесткого диска, либо с дискеты, вставленной в устройство A:, а если такой дискеты нет, то с жесткого диска. Возможны и другие случаи, здесь все зависит от конкретной реализации BIOS.

Итак, существуют две возможности загрузить операционную систему – с жесткого диска или с дискеты. Рассмотрим вначале первую возможность.

При загрузке с жесткого диска в память по фиксированному адресу читается содержимое главной загрузочной записи. Эта запись представляет собой программу, задачей которой является загрузка операционной системы с логического диска. Как эта загрузка выполняется?

Загрузчик, расположенный в главной загрузочной записи MBR, просматривает таблицу разделов диска Partition Table, которая находится в том же секторе диска, что и сама запись MBR. После того как в этой таблице будет найден раздел, отмеченный как активный, выполняется чтение самого первого сектора этого раздела в оперативную память – сектора загрузочной записи BR. В этом секторе находится еще один (!) загрузчик, на этот раз последний.

Задачей загрузчика BR является считывание в оперативную память стартовых модулей операционной системы и передача им управления. Способ загрузки, очевидно, зависит от операционной системы, поэтому каждая операционная система имеет свой собственный загрузчик BR.

Теперь о загрузке с дискеты. Этот процесс намного проще, так как формат дискеты в точности соответствует формату логического диска. Самый первый сектор нулевой дорожки дискеты содержит загрузочную запись BR, которая читается в память. После чтения ей передается управление. Заметим, что дискеты могут быть системными и несистемными.

Вы знаете, что системную дискету MS-DOS можно подготовить при помощи команды `format`, указав ей параметр `/s`, либо при помощи команды `sys`. И в том, и в другом случае в первый сектор нулевой дорожки дискеты записывается программа начальной загрузки MS-DOS.

Если же дискета была отформатирована командой `format` без параметра `/s`, она будет несистемной. Тем не менее, в первый сектор

нулевой дорожки дискеты все равно записывается программа, единственным назначением которой является вывод сообщения о необходимости вставить в НГМД системную дискету.

Данное обстоятельство – присутствие загрузочной записи на несистемной дискете – играет важную роль при распространении загрузочных вирусов, поэтому мы советуем обратить на него внимание.

Из сказанного выше следует, что загрузка операционной системы является многоступенчатым процессом, ход которого зависит от разных обстоятельств. Для нас сейчас важно то, что в этом процессе задействовано три программы, которые служат объектом нападения загрузочных вирусов:

- 1) главная загрузочная запись;
- 2) загрузочная запись на логическом диске;
- 3) загрузочная запись на дискете.

Вирусы могут заменять некоторые или все перечисленные выше объекты, встраивая в них свое тело и сохраняя содержимое оригинального загрузочного сектора в каком-либо более или менее подходящем для этого месте на диске компьютера. В результате при включении компьютера программа загрузки, расположенная в BIOS, загружает в память вирусный код и передает ему управление. Дальнейшая загрузка операционной системы происходит под контролем вируса, что затрудняет, а в некоторых случаях и исключает его обнаружение антивирусными программами.

Как распространяются загрузочные вирусы? Главным образом, с помощью забывчивых пользователей. Разработчик вируса создает дискету с зараженным загрузочным сектором или заражает главную загрузочную запись на жестком диске компьютера, к которому имеет доступ много пользователей. После загрузки операционной системы вирус контролирует все обращения к дискетам. Как только пользователь вставит дискету, не защищенную от записи, вирус запишет свое тело в загрузочный сектор дискеты. Через некоторое время все дискеты, которые когда-либо вставлялись в компьютер, окажутся зараженными.

Вставив зараженную дискету в устройство A: ранее незараженного компьютера и поработав с ней, многие пользователи забывают извлечь ее оттуда. Пользователь выключает компьютер и уходит домой спать, а вирус ждет своего часа. Включив утром компьютер, пользователь выполняет загрузку с забытой дискеты, в результате чего вирус проникает в главную загрузочную запись. Все, цель достигнута – вирус завоевал еще один компьютер.

Вы можете полностью перекрыть доступ к вашему компьютеру для загрузочных вирусов, отключив при помощи программы BIOS Setup возможность загрузки с устройства A:. Хотя иногда (например, при переустановке операционной системы) вам все же придется загружать компьютер с дискет.

Разумеется, не следует снимать с дискет защиту от записи без крайней на то необходимости. Особенно это относится к дистрибутив-

ным дискетам, с которых выполняется установка программного обеспечения. И конечно, все дискеты, полученные вами, следует проверять антивирусными программами. Это относится даже к новым форматированным дискетам в запечатанной коробке, так как вирус может находиться в области загрузочной записи. Известен случай, когда в продажу поступили зараженные форматированные дискеты.

КОМБИНИРОВАННЫЕ ФАЙЛОВО-ЗАГРУЗОЧНЫЕ ВИРУСЫ. Наиболее совершенные и наиболее опасные вирусы используют методы распространения, характерные и для файловых, и для загрузочных вирусов. Такие вирусы записывают свое тело в файлы и в загрузочные записи дискет и дисков. Вы можете получить такой вирус, загрузив компьютер с зараженной дискеты, либо запустив зараженный файл. Результат будет одинаковый – вирус поселится в вашем компьютере и начнет свою “работу”.

ПРОСТЫЕ И ПОЛИМОРФНЫЕ ВИРУСЫ. Обычные компьютерные вирусы обнаружить достаточно легко, так как в процессе заражения они записывают в заражаемый файл или системную область диска свой собственный код. Автору антивирусной программы достаточно выделить из этого кода уникальную последовательность команд или байт, характерную именно для данного вируса. Такая последовательность носит название сигнатуры.

Затем антивирусная программа уже в автоматическом режиме просматривает все файлы и системные области дисков в поиске сигнатур известных вирусов. Естественно, что проблем с обнаружением таких вирусов нет.

Очень скоро авторы вирусов догадались использовать в своих вирусах алгоритмы шифрования, затрудняющие их обнаружение и выделение сигнатуры. Такие вирусы, получившие название шифрующихся, при заражении новых файлов и системных областей диска шифруют собственный код, пользуясь для этого случайными паролями (ключами). Когда вирус получает управление, он первым делом расшифровывает собственный код.

Сложность обнаружения таких вирусов состоит в том, что код вируса случайным образом изменяется при каждом новом заражении и, соответственно, автору антивируса сложнее выделить сигнатуру такого вируса. Однако, так как шифрующийся вирус все же должен содержать неизменную процедуру расшифровки, то сигнатуру получить можно. Даже простые антивирусные программы способны успешно обнаруживать и удалять вирусы, применяющие алгоритм шифровки.

Вслед за шифрующимися вирусами появилась еще более сложная разновидность вирусов, получившая страшное название вирусом-мутантов. Более научное название вирусом-мутантов – полиморфные вирусы. От шифрующихся вирусов они отличаются тем, что даже процедура расшифровки меняется у разных особей одного вируса.

Каждый раз когда вирус заражает новый файл или системную область диска, он полностью изменяется, поэтому из полиморфных вирусов невозможно выделить сигнатуру.

Многие антивирусные программы не в состоянии обнаружить полиморфные вирусы. Например, самая известная антивирусная программа Aidtest, которая уже много лет защищает компьютеры, обнаруживает только самые примитивные экземпляры полиморфных вирусов. Мы видели много пользователей, постоянно проверяющих свои компьютеры программой Aidtest даже несмотря на предупреждение о необходимости дополнительно использовать антивирус Doctor Web. Полиморфные вирусы остаются в этом случае незамеченными и спокойно делают свое черное дело.

Извечная борьба щита и меча, брони и снаряда нашла свое отражение и в мире компьютеров. Для охоты за полиморфными вирусами были разработаны антивирусные программы нового поколения, далеко ушедшие от своих предшественников.

В качестве примеров программ, способных обнаруживать и удалять полиморфные вирусы, можно привести антивирус Doctor Web Игоря Данилова и Antiviral Toolkit Pro Евгения Касперского. Эвристический анализатор Doctor Web “выполняет” под своим управлением проверяемые программы и обнаруживает действия, характерные для вирусов. Благодаря этому он находит полиморфные вирусы так же легко, как и обычные вирусы, не использующие механизма маскировки.

Эвристический анализатор может обнаружить не только вирусы, ранее изученные автором антивирусной программы. Он может отыскать даже те вирусы, которые ранее не были известны. Надо сказать, что это не должно вызывать энтузиазм у авторов вирусов. Многие из них, еще не родившись, уже имеют все шансы быть обнаруженными.

СТЕЛС-ВИРУСЫ. В ходе проверки компьютера антивирусные программы считывают данные – файлы и системные области с жестких дисков и дискет, пользуясь средствами операционной системы и базовой системы ввода/вывода BIOS. Ряд вирусов после запуска оставляют в оперативной памяти компьютера специальные модули, перехватывающие обращение программ к дисковой подсистеме компьютера. Если такой модуль обнаруживает, что программа пытается прочитать зараженный файл или системную область диска, он на ходу подменяет читаемые данные, как будто вируса на диске нет.

Стелс-вирусы обманывают антивирусные программы и в результате остаются незамеченными. Тем не менее, существует простой способ отключить механизм маскировки стелс-вирусов. Достаточно загрузить компьютер с незараженной системной дискеты и сразу, не запуская других программ с диска компьютера (которые также могут оказаться зараженными), проверить компьютер антивирусной программой.

При загрузке с системной дискеты вирус не может получить управление и установить в оперативной памяти резидентный модуль,

реализующий стелс-механизм. Антивирусная программа сможет прочитать информацию, действительно записанную на диске, и легко обнаружит вирус.

Большинство антивирусных программ противодействуют попыткам стелс-вирусов остаться незамеченными, но чтобы не оставить им ни единого шанса, следует перед проверкой компьютера программами Antiviral Toolkit Pro, Aidstest и Doctor Web загружать компьютер с дискеты.

Системная дискета для антивирусного контроля должна быть подготовлена заранее. Кроме системных файлов, на нее следует записать антивирусные программы.

Многие антивирусные программы настолько успешно противостоят стелс-вирусам, что обнаруживают их при попытке замаскироваться. Такие программы считывают проверяемые программы с диска, пользуясь для этого различными методами. Например, с помощью операционной системы и базовой системы ввода/вывода. Если в полученных данных обнаруживается несоответствие, вероятнее всего, в оперативной памяти находится стелс-вирус.

2.4.3. Антивирусное ПО

Вне всякого сомнения, главным оружием в борьбе с вирусами являются антивирусные программы. Они позволяют не только обнаружить вирусы, в том числе вирусы использующие различные методы маскировки, но и удалить их из компьютера. Последняя операция может быть достаточно сложной и занять некоторое время. Так например, в последнее время широкое распространение получил вирус OneHalf – “одна половина”. Свое название он получил потому, что этот вирус постепенно зашифровывает информацию, записанную на жестком диске компьютера, а когда он зашифрует половину диска, то отображает соответствующую надпись на экране в момент загрузки компьютера. Вирус OneHalf весьма коварен. Пока он присутствует в компьютере, он не только постепенно шифрует данные на диске, он также расшифровывает их, когда какая-нибудь программа обращается к зашифрованным данным. Если вирус остается в компьютере, пользователь ничего не замечает. Если же удалить вирус, не расшифровав предварительно данные, записанные на диске, после очередной перезагрузки компьютера вы потеряете возможность их расшифровки.

В настоящее время насчитываются тысячи различных вирусов и десятки антивирусных программ для борьбы с ними. Наиболее известны следующие антивирусные программы для операционной системы MS-DOS: Aidstest, ADinf с ADinf Cure Module, Doctor Web, Scan, Vakcina, Antiviral Toolkit Pro и др. Проблема борьбы с вирусами стала настолько серьезной, что начиная с версии 6.0 операционная система MS-DOS включает собственные антивирусные средства: Microsoft Anti-Virus для MS-DOS и для Windows.

Существуют даже специальные программы для борьбы с вирусами

в сетях. Например, программа NetShield загружается на сервере Novell NetWare в качестве NLM-процесса. Она может выполнять, например, раз в сутки проверку дисков сервера на наличие вирусов. В случае обнаружения вирусов всем пользователям сети будет рассылаться соответствующее предупреждающее сообщение. Вы также можете задать режим, в котором NLM-антивирус будет проверять на заражение вирусом все новые файлы, записываемые на сервер.

Не остались в стороне и разработчики аппаратуры компьютеров. Некоторые версии BIOS современных системных плат позволяют отслеживать попытки записи в системные области жесткого диска. Если вирус нападет на загрузочный сектор или сектор начальной загрузки жесткого диска, установленного в таком компьютере, на экране появится предупреждающее сообщение.

Отечественное антивирусное программное обеспечение успешно конкурирует с программным обеспечением, предлагаемым зарубежными фирмами. Хорошая поддержка пользователей отечественных антивирусных программ со стороны фирм-разработчиков антивирусов позволяет своевременно получать новые версии антивирусов, а также консультации по их использованию.

Существует несколько основополагающих методов поиска вирусов, которые применяются антивирусными программами:

- сканирование;
- эвристический анализ;
- обнаружение изменений;
- резидентные мониторы.

Антивирусные программы могут реализовывать все перечисленные выше методики, либо только некоторые из них.

Сканирование. Сканирование является наиболее традиционным методом поиска вирусов. Оно заключается в поиске сигнатур, выделенных из ранее обнаруженных вирусов. Антивирусные программы-сканеры, способные удалить обнаруженные вирусы, обычно называются полифагами.

Недостатком простых сканеров является их неспособность обнаружить полиморфные вирусы, полностью меняющие свой код. Для этого необходимо использовать более сложные алгоритмы поиска, включающие эвристический анализ проверяемых программ.

Кроме того, сканеры могут обнаружить только уже известные и предварительно изученные вирусы, для которых была определена сигнатура. Поэтому программы-сканеры не защитят ваш компьютер от проникновения новых вирусов, которых, кстати, появляется по несколько штук в день. Как результат, сканеры устаревают уже в момент выхода новой версии.

Эвристический анализ. Эвристический анализ зачастую используется совместно со сканированием для поиска шифрующихся и полиморфных вирусов. В большинстве случаев эвристический анализ позволяет также обнаруживать и ранее неизвестные вирусы. В этом

случае, скорее всего, их лечение будет невозможно.

Если эвристический анализатор сообщает, что файл или загрузочный сектор возможно заражен вирусом, вы должны отнестись к этому с большим вниманием. Необходимо дополнительно проверить такие файлы с помощью самых последних версий антивирусных программ сканеров или передать их для исследования авторам антивирусных программ.

Обнаружение изменений. Заражая компьютер, вирус делает изменения на жестком диске: дописывает свой код в заражаемый файл, изменяет системные области диска и т. д. На обнаружении таких изменений основывается работа антивирусных программ-ревизоров.

Антивирусные программы-ревизоры запоминают характеристики всех областей диска, которые могут подвергнуться нападению вируса, а затем периодически проверяют их. В случае обнаружения изменений, выдается сообщение о том, что возможно на компьютер напал вирус.

Следует учитывать, что не все изменения вызваны вторжением вирусов. Так, загрузочная запись может измениться при обновлении версии операционной системы, а некоторые программы записывают внутри своего выполняемого файла данные.

Резидентные мониторы. Антивирусные программы, постоянно находящиеся в оперативной памяти компьютера и отслеживающие все подозрительные действия, выполняемые другими программами, носят название резидентных мониторов или сторожей. К сожалению, резидентные мониторы имеют очень много недостатков, которые делают этот класс программ малоприменимыми для использования. Они раздражают пользователей большим количеством сообщений, по большей части не имеющих отношения к вирусному заражению, в результате чего их отключают.

Рассмотрим наиболее распространенные программные средства, предназначенные для обнаружения и удаления вирусов – отечественную разработку – программу Aidstest, пакет Microsoft Anti-Virus, поставляемый в комплекте с операционной системой MS-DOS, и лучшую отечественную антивирусную программу Antiviral Toolkit Pro.

Программа Aidstest. Автором известной антивирусной программы Aidstest является Лозинский Дмитрий Николаевич. Aidstest позволяет не только обнаружить большое количество разнообразных вирусов. Эта программа также позволяет удалить обнаруженные вирусы.

Полное описание параметров программы Aidstest можно получить, запустив Aidstest с параметром /h. Мы позволим себе привести только самые главные параметры этой программы. Формат вызова Aidstest имеет следующий вид: AIDSTEST.EXE path [дополнительные параметры].

Первый, обязательный, параметр path должен задавать каталог в котором будет выполняться поиск вирусов. При этом автоматически проверяются все подкаталоги. Если в качестве первого параметра вы укажете только имя диска (например, AIDSTEST C:), то на наличие вирусов будет проверен весь диск.

Если вы не укажете программе Aidstest дополнительные параметры, то она будет осуществлять только поиск вирусов. В том случае, если вы желаете удалить найденные вирусы, вам необходимо задать дополнительный параметр /F. Aidstest будет исправлять зараженные вирусами программы и стирать не поддающиеся исправлению. Для того чтобы Aidstest запрашивал разрешение на удаление испорченных файлов программ, необходимо дополнительно задать параметр /Q.

Ниже представлен пример использования антивируса Aidstest для обнаружения и лечения вирусов на диске C: . AIDSTEST C: /F /Q.

Использование антивирусных программ не может гарантировать полную защиту компьютера от заражения вирусами. Дело в том, что антивирусы типа Aidstest настроены на обнаружение уже известных вирусов и их клонов. Новые вирусы могут быть неизвестны программе Aidstest и она в этом случае не сможет их обнаружить. Еще раз подчеркнем, что, на наш взгляд, наиболее надежным способом предохранить компьютер от нападения вирусов, является установка программных средств только с дистрибутивов и ограничение в использовании дискет для обмена программами.

Microsoft Anti-Virus. Программа Microsoft Anti-Virus включена в состав дистрибутива MS-DOS версии 6.0. При этом Microsoft Anti-Virus представлен двумя версиями – для MS-DOS и для Windows. Подробно процедуры установки и использования этих антивирусных программ описаны во втором томе серии “Персональный компьютер. Шаг за шагом”.

Antiviral Toolkit Pro (AVP). Эта программа – новый шаг в борьбе с компьютерными вирусами. Она представляет из себя полностью 32-разрядное приложение, оптимизированное для работы в среде Microsoft Windows 95 (Windows NT) и использующее все ее возможности. AVP имеет удобный пользовательский интерфейс, характерный для Windows 95, большое количество настроек, выбираемых пользователем, а также одну из самых больших в мире антивирусных баз, что гарантирует вам надежную защиту от огромного числа самых разнообразных вирусов.

В ходе работы AVP сканирует:

оперативную память (DOS, XMS, EMS);

файлы, включая архивные и упакованные;

системные сектора, содержащие Master Boot Record, загрузочный сектор (Boot-сектор) и таблицу разбиения диска (Partition Table).

К основным особенностям AVP относятся:

1. Детектирование и удаление огромного числа самых разнообразных вирусов, в том числе:
 - полиморфных или самошифрующихся вирусов;
 - стелс-вирусов или вирусов-невидимок;
 - новых вирусов для Windows 3.XX и Windows 95;
 - макровирусов, заражающих документы Word и таблицы Excel.
2. Сканирование внутри упакованных файлов (модуль Unpacking Engine).

3. Сканирование внутри архивных файлов (модуль Extracting Engine).
4. Сканирование объектов на гибких, локальных, сетевых и CD-ROM дисках.
5. Эвристический модуль Code Analyzer, необходимый для детектирования неизвестных вирусов.
6. Поиск в режиме избыточного сканирования.
7. Проверка объектов на наличие в них изменений.

В комплект поставки входит также AVP Monitor – резидентный модуль, находящийся постоянно в оперативной памяти компьютера и отслеживающий все файловые операции в системе. Он позволяет обнаружить и удалить вирус до момента реального заражения системы в целом.

Программа AVP имеет удобный пользовательский интерфейс, позволяет создавать, сохранять и загружать большое количество различных настроек, имеет мощный механизм проверки целостности антивирусной системы и мощную систему помощи. Для обновления базы данных вирусов нет необходимости переустанавливать программу – достаточно подключить файл обновления.

2.5. Защита от несанкционированного доступа

В вычислительных сетях сосредотачивается информация, исключительное право на пользование которой принадлежит определенным лицам или группам лиц, действующим в порядке личной инициативы или в соответствии с должностными обязанностями. Такая информация должна быть защищена от всех видов постороннего вмешательства: чтения лицами, не имеющими права доступа к информации, и преднамеренного изменения информации (т.е. от **несанкционированного доступа** – неправомерного обращения к ресурсам вычислительной системы с целью их непредусмотренного использования и (или) порчи). К тому же в вычислительных сетях должны приниматься меры по защите вычислительных ресурсов сети от их несанкционированного использования, т.е. должен быть исключен доступ к сети лиц, не имеющих на это права. Физическая защита системы и данных может осуществляться только в отношении рабочих ЭВМ и узлов связи и оказывается невозможной для средств передачи, имеющих большую протяженность. По этой причине в вычислительных сетях должны использоваться средства, исключающие несанкционированный доступ к данным, обеспечивающие их секретность и **безопасность** работы (способность системы обработки данных обеспечить защиту, достоверность и надежность хранения информации).

Неизбежным следствием борьбы с этой опасностью стали постоянно увеличивающиеся расходы на защиту информации. Например, по оценке немецких экспертов, лишь в 1987 г. в промышленности и учебных заведениях Западной Европы потрачено 1,7 млрд. марок на обеспечение безопасности компьютеров.

Под **защитой** понимаются средства для ограничения доступа или использования всей вычислительной системы или ее части. Разделяют **защиту данных** (организационные, программные и технические методы и средства, направленные на удовлетворение ограничений, установленных для типов данных или экземпляров типов данных в системе обработки данных), **защиту памяти** (средства и способы предотвращения несанкционированного доступа к определенным участкам памяти с целью сохранения неизменности записанных данных при появлении неисправностей в оборудовании, некорректных действиях оператора или вследствие ошибок в программе) и **защиту от ошибок** (средства и способы контроля допустимости входных параметров).

Исследования практики функционирования систем обработки данных и вычислительных систем показали, что существует достаточно много возможных направлений утечки информации и путей несанкционированного доступа в системах и сетях. В их числе:

- чтение остаточной информации в памяти системы после выполнения санкционированных запросов; копирование носителей информации и файлов информации определением мер защиты;
- маскировка под зарегистрированного пользователя;
- маскировка под запрос системы;
- использование программных ловушек;
- использование недостатков операционной системы;
- незаконное подключение к аппаратуре и линиям связи;
- злоумышленный вывод из строя механизмов защиты;
- внедрение и использование компьютерных вирусов.

Для защиты вычислительной сети от несанкционированного доступа применяется идентификация пользователей (сообщений), позволяющая устанавливать конкретного пользователя, работающего за терминалом и принимающего либо отправляющего сообщения.

Право доступа к определенным вычислительным и информационным ресурсам, программам и наборам данных, а также вычислительной сети в целом предоставляется ограниченному контингенту лиц, и система должна идентифицировать пользователей, работающих за терминалами. Идентификация пользователей чаще всего производится с помощью паролей – это так называемая **защита паролем** (средства, предотвращающие доступ посторонних лиц к запрещенным данным, которые недоступны для обработки, пока оператор не введет требуемый пароль). *Пароль* – совокупность символов, известных подключенному к сети абоненту, вводится им в начале сеанса взаимодействия с сетью, а иногда и в конце сеанса (в особо ответственных случаях пароль нормального выхода из сети может отличаться от входного). Наконец, система может предусматривать ввод пароля для подтверждения правомочности пользователя через определенные кванты времени. Вычислительная система определяет подлинность пароля и тем самым пользователя.

Для защиты средств идентификации пользователей от неправомерного использования пароли передаются и сравниваются в зашифрованном виде, а таблицы паролей также хранятся в зашифрованном виде, что исключает возможность прочтения паролей без знания ключей.

Для идентификации пользователей могут использоваться и физические методы, например карточка с магнитным покрытием, на которой записывается персональный идентификатор пользователя, карточки с встроенным чипом. Для уменьшения риска злоупотреблений, как правило, используются карточки с каким-либо другим способом идентификации пользователя, например с коротким паролем.

Наиболее надежным (хотя и наиболее сложным) является способ идентификации пользователя на основе анализа его индивидуальных параметров: отпечатков пальцев, рисунка линий руки, радужной оболочки глаз и др.

Еще один способ защиты от несанкционированного доступа – использование **авторизованных программ** (программ пользователя, имеющих ограничения по доступу к ним со стороны других пользователей) и ограничение доступа к файлам и директориям. Так, например, в операционной системе Windows NT каждый пользователь имеет доступ лишь к разрешенным для него программам, а также файлам и директориям.

Во многих современных прикладных программах уже предусмотрены свои меры защиты информации. Например, в Microsoft Access помимо установки пароля, требуемого при открытии базы данных, предусмотрена и защита на уровне пользователей, которая позволяет ограничить, к какой части базы данных пользователь будет иметь доступ или какую ее часть он сможет изменять. Кроме того, можно удалить изменяемую программу Visual Basic из базы данных, чтобы предотвратить изменения структуры форм, отчетов и модулей, сохранив базу данных как файл MDE.

Простейшим способом защиты является установка пароля для открытия базы данных. После того как пароль установлен, при каждом открытии базы данных будет появляться диалоговое окно, в которое требуется ввести пароль. Только те пользователи, которые введут правильный пароль, смогут открыть базу данных. Этот способ достаточно надежен (Microsoft Access шифрует пароль, так что к нему нет прямого доступа при чтении файла базы данных), но он применяется только при открытии базы данных. После открытия базы данных все объекты становятся доступными для пользователя (пока не определена защита на уровне пользователей). Для базы данных, которая совместно используется небольшой группой пользователей, или на автономном компьютере установка пароля обычно оказывается достаточной. Однако нельзя использовать пароль базы данных, если предполагается выполнять её репликацию, т.к. реплицированные базы данных не могут быть синхронизированы, если определен пароль базы данных.

Наиболее гибкий и распространенный способ защиты базы данных называется защитой на уровне пользователей. Этот способ защиты

подобен способам, используемым в большинстве сетевых систем. От пользователей требуется идентифицировать себя и ввести пароль, когда они запускают Microsoft Access. Внутри файла рабочей группы они идентифицируются как члены группы. Microsoft Access по умолчанию создает две группы: администраторы (группа <Admins>) и пользователи (группа <Users>). Допускается также определение других групп.

Группам и пользователям предоставляются разрешения на доступ, определяющие возможность их доступа к каждому объекту базы данных. Например, члены группы <Users> могут иметь разрешения на просмотр, ввод или изменение данных в таблице <Клиенты>, но им не будет разрешено изменять структуру этой таблицы. Группа <Users> может быть допущена только к просмотру данных в таблице, содержащей сведения о заказах, и не иметь доступа к таблице <Платежная ведомость>. Члены группы <Admins> имеют все разрешения на доступ ко всем объектам базы данных. Имеется возможность установить более разветвленную структуру управления, создавая собственные учетные записи групп, предоставляя этим группам соответствующие разрешения и добавляя в них пользователей.

Следует отметить три главных преимущества защиты на уровне пользователей:

1. Защищается ваша программа как интеллектуальная собственность.
2. Защищается приложение от повреждения из-за неумышленного изменения пользователями программ или объектов, от которых зависит работа приложения.
3. Защищаются конфиденциальные сведения в базе данных.

2.6. Технологическая безопасность

Обеспечение безопасности информации в вычислительных сетях в автономно работающих ЭВМ достигается комплексом организационных, организационно-технических и программных мер (все вместе это называется технологической безопасностью). К *организационным мерам защиты* относятся:

ограничение доступа в помещения, в которых происходит подготовка и обработка информации;

допуск к обработке и передаче конфиденциальной информации только проверенных должностных лиц;

хранение магнитных носителей и регистрационных журналов в сейфах, закрытых для доступа посторонних лиц;

исключение просмотра посторонними лицами содержания обрабатываемых материалов через дисплей, принтер и т.д.;

использование криптографических кодов при передаче по каналам связи ценной информации;

уничтожение красящих лент, бумаги и иных материалов, содержащих фрагменты ценной информации.

Организационно-технические меры включают:

осуществление питания оборудования, обрабатывающего ценную информацию от независимого источника питания или через специальные сетевые фильтры;

установку на дверях помещений кодовых замков;

использование для отображения информации при вводе-выводе жидкокристаллических или плазменных дисплеев, а для получения твердых копий – струйных принтеров и термопринтеров, поскольку дисплей дает такое высокочастотное электромагнитное излучение, что изображение с его экрана можно принимать на расстоянии нескольких сотен километров;

уничтожение информации, хранящейся в ПЗУ и на НЖМД, при списании или отправке ПЭВМ в ремонт;

установку клавиатуры и принтеров на мягкие прокладки с целью снижения возможности снятия информации акустическим способом;

ограничение электромагнитного излучения путем экранирования помещений, где проходит обработка информации, листами из металла или из специальной пластмассы.

Технические средства защиты – это системы охраны территорий и помещений с помощью экранирования машинных залов и организации контрольно-пропускных систем.

Защита информации в сетях и вычислительных средствах с помощью технических средств реализуется на основе организации доступа к памяти с помощью:

контроля доступа к различным уровням памяти компьютеров;

блокировки данных и ввода ключей;

выделения контрольных битов для записей с целью идентификации и др.

Архитектура программных средств защиты информации включает:

контроль безопасности, в том числе контроль регистрации вхождения в систему, фиксацию в системном журнале, контроль действий пользователя;

реакцию (в том числе звуковую) на нарушение системы защиты контроля доступа к ресурсам сети;

контроль мандатов доступа;

формальный контроль защищенности операционных систем (базовой общесистемной и сетевой);

контроль алгоритмов защиты;

проверку и подтверждение правильности функционирования технического и программного обеспечения.

Для надежной защиты информации и выявления случаев неправомерных действий проводится регистрация работы системы: создаются специальные дневники и протоколы, в которых фиксируются все действия, имеющие отношение к защите информации в системе. Фиксируются время поступления заявки, ее тип, имя пользователя и терминала, с которого инициализируется заявка. При отборе событий,

подлежащих регистрации, необходимо иметь в виду, что с ростом количества регистрируемых событий затрудняется просмотр дневника и обнаружение попыток преодоления защиты. В этом случае можно применять программный анализ и фиксировать сомнительные события.

Используются также специальные программы для тестирования системы защиты. Периодически или в случайно выбранные моменты времени они проверяют работоспособность аппаратных и программных средств защиты.

К отдельной группе мер по обеспечению сохранности информации и выявлению несанкционированных запросов относятся программы обнаружения нарушений в режиме реального времени. Программы данной группы формируют специальный сигнал при регистрации действий, которые могут привести к неправомерным действиям по отношению к защищаемой информации. Сигнал может содержать информацию о характере нарушения, месте его возникновения и другие характеристики. Кроме того, программы могут запретить доступ к защищаемой информации или создать такой режим работы (например, моментальная загрузка устройств ввода-вывода), который позволит выявить нарушителя и задержать его соответствующей службой.

Один из распространенных способов защиты – явное указание секретности выводимой информации. В системах, поддерживающих несколько уровней секретности, вывод на экран терминала или печатающего устройства любой единицы информации (например, файла, записи или таблицы) сопровождается специальным грифом с указанием уровня секретности. Это требование реализуется с помощью соответствующих программных средств.

В отдельную группу выделены средства защиты от несанкционированного использования программного обеспечения. Они приобретают особое значение вследствие широкого распространения персональных компьютеров. Исследования, проведенные зарубежными экспертами, свидетельствуют, что на одну проданную копию оригинальной программы приходится минимум одна нелегальная копия. А для особо популярных программ это соотношение достигает 1:7.

Особое внимание уделяется законодательным средствам, регулирующим использование программных продуктов. В соответствии с Законом Российской Федерации об информации и информатизации от 25 января 1995 г. предусматриваются санкции к физическим и юридическим лицам за нелегальное приобретение и использование программных средств.

Большую опасность представляют компьютерные вирусы. Программа, внутри которой находится компьютерный вирус, называется зараженной. Когда такая программа начинает работу, то сначала управление получает вирус, который находит и заражает другие программы, а также выполняет ряд вредных действий, в частности, «засоряет» активную память, портит файлы и т.д. (подробнее об этом написано выше).

Если не принимать мер по защите от вируса, то последствия заражения вирусом компьютеров могут быть серьезными. В число средств и методов защиты от компьютерных вирусов входят:

- общие средства защиты информации, которые полезны так же, как и страховка от физической порчи машинных дисков, неправильно работающих программ или ошибочных действий пользователя;

- профилактические меры, позволяющие уменьшить вероятность заражения вирусом;

- специализированные программы для защиты от вирусов.

Комплексное решение вопросов безопасности ВС принято именовать *архитектурой безопасности*, где выделяются угрозы безопасности, службы безопасности и механизмы обеспечения безопасности. Под *угрозой безопасности* понимается действие или событие, которое может привести к разрушению, искажению или несанкционированному использованию ресурсов сети, включая хранимую и обрабатываемую информацию, а также программные и аппаратные средства.

Угрозы подразделяются на случайные (непреднамеренные) и умышленные. Источником первых могут быть ошибки в ПО, неправильные (ошибочные) действия пользователей, выход из строя аппаратных средств и др.

Умышленные угрозы преследуют цель нанесения ущерба пользователям (абонентам) ВС и подразделяются на активные и пассивные. Пассивные угрозы не разрушают информационные ресурсы и не оказывают влияния на функционирование ВС. Их задача – несанкционированно получать информацию. Активные угрозы преследуют цель нарушать нормальный процесс функционирования ВС путем разрушения или радиоэлектронного подавления линий связи ВС, вывода из строя ЭВМ или ее ОС, искажения баз данных и т.д. Источником активных угроз могут быть непосредственные действия физических лиц, злоумышленников, компьютерные вирусы и т.д.

К основным угрозам безопасности относятся: раскрытие конфиденциальной информации, несанкционированное использование ресурсов ВС, отказ от информации.

Создаваемая служба безопасности вычислительной сети призвана обеспечивать:

- подтверждение подлинности того, что объект, который предлагает себя в качестве отправителя информации в сети, действительно им является;

- целостность информации, выявляя искажения, вставки, повторы и уничтожение данных, передаваемых в сетях, а также последующее восстановление данных;

- секретность всех данных, передаваемых по каналам ВС;

- нейтрализацию попыток несанкционированного использования ресурсов ЭВМ. При этом контроль доступа может быть либо избирательным, т.е. распространяться только на некоторые виды доступа

к ресурсам, например, на обновление информации в базе данных, либо полным;

нейтрализацию угрозы отказа от информации со стороны ее отправителя и/или получателя;

получателя информации доказательствами, которые исключают попытки отправителя отрицать факты передачи указанной информации или ее содержания.

К механизмам обеспечения безопасности относятся: идентификация пользователей (рассмотрена выше), шифрование данных, электронная подпись, управление маршрутизацией и др.

Рассмотрим некоторые из них.

Шифрование данных. Секретность данных обеспечивается методами криптографии, т.е. методами преобразования данных из общепринятой формы в кодированную (шифрование) и обратного преобразования (дешифрование) на основе правил, известных только взаимодействующим абонентам сети. Криптография применяется для защиты передаваемых данных, а также информации, хранимой в базах данных: в пакетах дисков, на гибких дисках и лентах.

Шифрование данных производится по алгоритму, определяющему порядок преобразования исходного текста в зашифрованный текст, дешифрование – по алгоритму, реализующему обратное преобразование. К криптографическим средствам предъявляются требования хранения секретности, даже когда известна сущность алгоритмов шифрования-дешифрования. Секретность обеспечивается введением в алгоритмы специальных ключей (кодов). Зашифрованный текст превращается в исходный только в том случае, когда и в процессе шифрования и дешифрования использовался один и тот же ключ. Область значений ключа выбирается столь большой, что практически исключается возможность его определения путем простого перебора возможных значений.

Таким способом вы защитите вашу конфиденциальную информацию, даже если в вашу сеть произойдет проникновение, или ваш системный администратор захочет причинить вам вред. Шифрование также важно в том случае, если вы используете компьютер совместно с другими пользователями.

Если вы регулярно делаете резервные копии данных и используете шифрование, вы можете спокойно пользоваться компьютером совместно с другими. Перед тем, как шифровать ваши файлы, вы должны проверить, можно ли это делать согласно политике безопасности вашей организации. Некоторые работодатели и страны явно запрещают хранение зашифрованных файлов или передачу зашифрованных файлов по сети.

Необходимо быть осторожным с паролями или ключами для шифрования файлов. Хранение их в безопасном месте не только поможет уберечь их, но также поможет вам сохранить в безопасности ваши данные; так как, если вы потеряете пароли или ключи, вы не

сможете расшифровать ваши данные. Иногда разумно иметь несколько копий паролей или ключей. Это требование может оказаться обязательным, если ваша организация, например, имеет политику расщепленных ключей. В этом случае обеспечивается защита от того, что кто-то из сотрудников по тем или иным причинам может уйти из компании и унести с собой пароль.

Хотя сейчас можно воспользоваться большим числом программ для шифрования, качество их может резко отличаться. Например, программа PGO обеспечивает хорошие возможности шифрования. Возможности шифрования также включены в ряд других программ (например, в программы Microsoft Office), но в этом случае качество шифрования очень низкое.

Многие сетевые службы требуют регистрации (log in). Пользователю на экран выдается предложение ввести имя своего аккаунта (username или account ID) и пароль. Если эта информация посылается через сеть в нешифрованном виде, это сообщение может быть перехвачено кем-либо. Правда, этого не происходит при подключении к сети через телефонную сеть, при котором вы устанавливаете соединение с сетью и вход в нее по коммутируемому каналу, так как перехватить данные, передаваемые по телефонной линии гораздо труднее, чем перехватить данные, передаваемые по сети.

Вы подвергаетесь такому риску тогда, когда вы используете какие-либо программы для подключения к удаленным компьютерам в сети. Многие из распространенных программ для подключения к компьютерам или для передачи файлов между компьютерами (например, telnet и ftp) посылают ваше имя и пароль, а также ваши данные по сети в открытом виде.

Обычной мерой предосторожности, предпринимаемой в организациях для защиты от перехвата пароля, является использование систем с одноразовыми паролями. До недавнего времени эти системы были слишком сложны и слишком дороги для простых пользователей и маленьких организаций. Но все увеличивающееся число программ этого типа уже сейчас позволяет реализовать такие системы не на базе дорогого оборудования, а на основе криптографических технологий. Примером такой технологии является программа Secure Shell (SSH) разных платформ. Многие продукты (в том числе и SSH), также позволяют шифровать все данные, передаваемые по сети.

Самый большой риск при работе в WWW – это загрузка на ваш компьютер файлов. WWW-браузеры позволяют загрузить любой файл из Интернет. WWW-браузеры загружают файлы даже в тех случаях, когда это не совсем очевидно. Поэтому риск, возникающий при загрузке файлов, может иметь место даже тогда, когда вы сами ничего явно не загружаете из Интернет. WWW-страницы часто включают формы. Как и в случае с электронной почтой, посылка данных вашим браузером WWW-серверу небезопасна. Имеется несколько способов для защиты от этого, самым важным из них является Secure Sockets Layer (SSL). Это средство

встроено во многие WWW-браузеры. С его помощью все сообщения, посылаемые между WWW-браузером пользователя и WWW-сервером шифруются, чтобы никто по пути не смог прочесть их.

Противостояние атакам — важное свойство защиты. Казалось бы, если в сети установлен межсетевой экран (firewall), то безопасность гарантирована, но это распространенное заблуждение может привести к серьезным последствиям. Например, межсетевой экран не способен защитить от пользователей, прошедших аутентификацию. Кроме того, межсетевой экран не только не защищает от проникновения в сеть через модем или иные удаленные точки доступа, но и не может обнаружить такого злоумышленника.

При этом система защиты, созданная на основе модели адаптивного управления безопасностью сети (Adaptive Network Security, ANS), способна решить все или почти все перечисленные проблемы. Она позволяет обнаруживать атаки и реагировать на них в режиме реального времени, используя правильно спроектированные, хорошо управляемые процессы и средства защиты.

Компания Yankee Group опубликовала в июне 1998 г. отчет, содержащий описание процесса обеспечения адаптивной безопасности сети. Этот процесс должен включать в себя анализ защищенности (security assessment), т. е. поиск уязвимостей (vulnerabilities assessment), обнаружение атак (intrusion detection), а также использовать адаптивный (настраиваемый) компонент, который расширяет возможности двух первых функций, и управляющий компонент.

Анализ защищенности осуществляется на основе поиска уязвимых мест во всей сети, состоящей из соединений, узлов (например, коммуникационного оборудования), хостов, рабочих станций, приложений и баз данных. Эти элементы нуждаются как в оценке эффективности их защиты, так и в поиске в них неизвестных уязвимостей. Процесс анализа защищенности предполагает исследование сети для выявления в ней «слабых мест» и обобщение полученных сведений, в том числе в виде отчета. Если система, реализующая данную технологию, содержит адаптивный компонент, то устранение найденной уязвимости будет осуществляться автоматически. При анализе защищенности обычно идентифицируются:

- «люки» в системах (back door) и программы типа «троянский конь»;
- слабые пароли;
- восприимчивость к проникновению из внешних систем и атакам типа «отказ в обслуживании»;
- отсутствие необходимых обновлений (patch, hotfix) операционных систем;
- неправильная настройка межсетевых экранов, Web-серверов и баз данных.

К сожалению, эффективно реализовать все описанные технологии в одной системе пока не удастся, поэтому пользователям приходится применять совокупность систем защиты, объединенных единой

концепцией безопасности. Пример таких систем — семейство продуктов SAFEsuite, разработанных американской компанией Internet Security Systems (ISS). Сегодня это — единственный комплект средств, который включает в себя все компоненты модели адаптивного управления защиты сети.

Сначала в него входили всего три продукта: система анализа защищенности на уровне сети Internet Scanner, средства анализа защищенности на уровне хоста System Scanner и обнаружения атак на уровне сети RealSecure Network Engine. В дальнейшем ISS дополнила комплект системой анализа защищенности на уровне баз данных Database Scanner и средствами обнаружения атак на уровне хоста RealSecure System Agent.

В настоящее время комплект ПО SAFEsuite поставляется в новой версии — SAFEsuite Enterprise, в которую входит также ПО SAFEsuite Decisions, обеспечивающее принятие решений по проблемам безопасности.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

- 1. Составьте логическую схему базы знаний по теме юниты.*

ТРЕНИНГ УМЕНИЙ

Пример выполнения упражнения на умение № 1

Задание

Добавить в редактируемый проект MyProject.dpr существующий модуль 2.pas.

Решение

№ п/п	Алгоритм	Конкретное соответствие данной ситуации предложенному алгоритму
1.	Открыть проект	<i>Из исходных данных следует, что проект MyProject.dpr уже открыт.</i>
2.	Открыть менеджер проекта (из меню «View» пункт «Project Manager»)	<i>Появляется окно «Project Manager», в котором виден список всех модулей, входящих в проект.</i>
3.	В окне «Project Manager» нажать кнопку «Add»	<i>Появляется окно «Открыть» («Open») со списком файлов в определенной директории.</i>
4.	Найти на диске файл 2.pas	<i>Из списка файлов в директории выбрать файл 2.pas.</i>
5.	Нажать кнопку «ОК»	<i>В окне «Project Manager» в список модулей добавляется файл 2.pas.</i>
6.	Заккрыть окно «Project Manager»	<i>Окно «Project Manager» закрывается.</i>

Решите самостоятельно следующие задания:

Задание 1

Добавить в проект MyProject.dpr новый модуль с именем NewUnit.pas.

Задание 2

Удалить из проекта MyProject.dpr модуль с именем MyUnit.pas.

Задание 3

Удалить из проекта MyProject.dpr модуль с именем MyUnit.pas и добавить новый модуль с именем NewUnit.pas.

Пример выполнения упражнения на умение № 2

Задание

Имеется следующая ассемблерная программа, сдвигающая содержимое ячеек памяти TMP1, TMP2 и TMP3 на 2 бита вправо:

```
...  
ASR TMP1  
ASR TMP1  
ASR TMP2  
ASR TMP2  
ASR TMP3  
ASR TMP3  
...
```

Требуется написать макрокоманду и вставить в программу ее макровывоз.

Решение

№ п/п	Алгоритм	Конкретное соответствие данной ситуации предложенному алгоритму
1.	Создать заголовок макрокоманды	.SHIFT2R W1,W2,W3
2.	Описать тело макро- команды	ASR W1 ASR W1 ASR W2 ASR W2 ASR W3 ASR W3
3.	Определить конец макрокоманды	.ENDM
4.	Определить в пр о- грамме макр овызов	... SHIFT2R TMP1, TMP 2, TMP 3 ...

Решите самостоятельно следующие задания:

Задание 1

Имеется следующая ассемблерная программа, содержащая в себе много одинаковых фрагментов кода (копирование содержимого регистров R0 и R1 в содержимое ячеек памяти TMP1, TMP2):

```
...
MOV R0, TMP1
MOV R1, TMP2
```

Требуется написать макрокоманду с двумя формальными параметрами и вставить в программу ее макровывоз.

Задание 2

Имеется следующая ассемблерная программа, содержащая в себе много одинаковых фрагментов кода (копирование содержимого регистра R0 в регистр R1):

```
...  
MOV R0, R1  
...
```

Требуется написать макрокоманду без формальных параметров и вставить в программу ее макровывоз.

Задание 3

Имеется следующая ассемблерная программа, содержащая в себе много одинаковых фрагментов кода (сдвигающая содержимое ячеек памяти на 2 бита вправо):

```
...  
ASR T1  
ASR T1  
...  
ASR T3  
ASR T3  
...
```

Требуется написать макрокоманду с одним формальным параметром и вставить в программу ее макровывоз.

Пример выполнения упражнения на умение № 3

Задание

Выбрать интерфейсы для многомашинной ЛВС, расположенной в пределах нескольких смежных комнат (расстояние между ЭВМ менее 100 м). Из периферийных устройств используются принтеры.

Решение

№ п/п	Алгоритм	Конкретное соответствие данной ситуации предложенному алгоритму
1.	Определить тип вычислительной системы	<i>Согласно исходных данных вычислительная система является многомашинной.</i>
2.	Определить расстояние между ЭВМ	<i>Согласно исходных данных расстояние между ЭВМ не превышает 100 м.</i>
3.	Определить типы периферийных устройств	<i>Из исходных данных следует, что периферийными устройствами являются принтеры.</i>
4.	Выбор вида интерфейса	<i>Из исходных данных и обеспечения максимальной скорости передачи данных для общения между ЭВМ необходимо выбрать параллельный интерфейс, а для периферийных устройств – последовательный.</i>

Решите самостоятельно следующие задания:

Задание 1

Выбрать интерфейсы для многомашинной ЛВС, расположенной в нескольких соседних зданиях (расстояние между ЭВМ менее 10-1000 м). Периферийные устройства не используются.

Задание 2

Выбрать интерфейсы для распределенной ЛВС, узлы которой расположены в нескольких соседних зданиях (расстояние между ЭВМ менее 10-1000 м), а расстояние между узлами более 5 км. Периферийные устройства не используются.

Пример выполнения упражнения на умение № 4

Задание

Выбрать канал связи, обеспечивающий максимально быструю доставку сообщений. Канал связи предполагается использовать периодически, режим передачи прерывистый.

Решение

№ п/п	Алгоритм	Конкретное соответствие данной ситуации предложенному алгоритму
1.	Определить режим работы канала	<i>Согласно исходных данных канал связи является коммутируемым.</i>
2.	Определить режим передачи	<i>Согласно исходных данных канал связи прерывистый.</i>
3.	Определить необходимую быстроту передачи сообщения	<i>Из исходных данных следует, что скорость передачи должна быть максимальной – выбираем канал связи с коммутацией пакетов.</i>

Решите самостоятельно следующие задания:

Задание 1

Выбрать канал связи, обеспечивающий максимально быструю доставку сообщений больших объемов. Загрузка канала связи предполагается максимальной.

Задание 2

Выбрать канал связи, обеспечивающий непрерывную доставку сообщений. Канал связи предполагается использовать периодически.

Задание 3

Выбрать канал связи, обеспечивающий доставку сообщений (не обязательно быструю) с максимальной эффективностью использования канала. Загрузка канала связи – по необходимости.

Пример выполнения упражнения на умение № 5

Задание

Выбрать физический носитель сигналов в ЛВС, обеспечивающий высокую скорость передачи информации и обладающий высокой степенью защиты информации.

Решение

№ п/п	Алгоритм	Конкретное соответствие данной ситуации предложенному алгоритму
1.	Выбрать физический носитель сигналов в ЛВС исходя из скорости передачи данных	<i>Согласно исходных данных скорость передачи высокая. Согласно пункта 1.2 алгоритма выбираем коаксиальный кабель.</i>
2.	Выбрать физический носитель сигналов в ЛВС исходя из степени защиты данных	<i>Согласно исходных данных степень защиты данных должна быть высокой, значит выбираем оптоволоконный кабель.</i>
3.	Сделать окончательный выбор	<i>Согласно пунктов 1 и 2 – окончательный выбор - оптоволоконный кабель.</i>

Решите самостоятельно следующие задания:

Задание 1

Выбрать физический носитель сигналов в ЛВС, обеспечивающий высокую скорость передачи информации, обладающий средней степенью защиты информации и минимальной стоимостью.

Задание 2

Выбрать физический носитель сигналов в ЛВС, обладающий минимальной стоимостью при средней степени защиты информации.

Задание 3

Выбрать физический носитель сигналов в ЛВС, обладающий минимальной стоимостью.

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

ЮНИТА 2

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРОЦЕССОВ РАЗРАБОТКИ И СОПРОВОЖДЕНИЯ ПРОГРАММНЫХ КОМПЛЕКСОВ

Редактор Л.С. Лебедева
Оператор компьютерной верстки Д.В. Федотов

Изд. лиц. ЛР № 071765 от 07.12.1998	Сдано в печать
НОУ "Современный Гуманитарный Институт"	
Тираж	Заказ
