**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ**

**Классификация программного обеспечения. Виды программного**

**обеспечения**

Программные средства для компьютеров можно разделить на:

– *прикладные программы*, непосредственно обеспечивающие выполнение необходимых пользователям работ;

– *системные программы*, выполняющие различные вспомогательные функции, например: управление ресурсами компьютера, создание копий используемой информации, проверка работоспособности устройств компьютера, выдача справочной информации о компьютере и др.;

– *инструментальные программные системы*, облегчающие процесс создания новых программ для компьютера.

При построении классификации ПО нужно учитывать тот факт, что стремительное развитие вычислительной техники и расширение сферы приложения компьютеров резко ускорили процесс эволюции программного обеспечения.

**Операционные системы: понятие и назначение**

Операционная система – это комплекс программ, обеспечивающих:

* управление ресурсами, т.е. согласованную работу всех аппаратных средств компьютера;
* управление процессами, т.е. выполнение программ, их взаимодействие с устройствами компьютера, с данными;
* пользовательский интерфейс, т.е. диалог пользователя с компьютером, выполнение определенных простых команд – операций по обработке информации.

Операционные системы – наиболее машиннозависимый вид программного обеспечения, ориентированный на конкретные модели компьютеров, поскольку они напрямую управляют их устройствами или, как еще говорят, обеспечивают интерфейс между пользователем и аппаратной частью компьютера.

За время существования компьютеров операционные системы претерпели значительную эволюцию. Так, первые операционные системы были *однопользовательскими и однозадачными*. Эффективность использования ресурсов компьютера в этом случае оказывалось невысокой из-за простоев всех, кроме одного работающего периферийного устройств компьютера. Например, при вводе данных простаивал центральный процессор, устройства вывода и внешние запоминающие устройства.

По мере роста возможностей, производительности и изменениях в соотношении стоимости устройств компьютера положение стало нетерпимым, что привело к появлению *многозадачных* операционных систем, остававшихся однопользовательскими.

Такие операционные системы обеспечивают постановку заданий в очередь на выполнение, *параллельное выполнение* заданий, *разделение ресурсов* компьютера между выполняющимися заданиями. Так, например, одно задание может выполнять ввод данных, другое – выполняться центральным процессором, третье – выводить данные, четвертое – стоять в очереди. Важнейшее техническое решение, обусловившее такие возможности, – появление у внешних устройств собственных процессоров (контроллеров).

При многозадачном режиме

* в оперативной памяти находится несколько заданий пользователей;
* время работы процессора разделяется между программами, находящимися в оперативной памяти и готовыми к обслуживанию процессором;
* параллельно с работой процессора происходит обмен информацией с различными внешними устройствами.

Наиболее совершенны и сложны *многопользовательские многозадачные* операционные системы, которые предусматривают одновременное выполнение многих заданий многих пользователей, обеспечивают *разделение ресурсов* компьютера в соответствии с приоритетами пользователей и *защиту данных* каждого пользователя от несанкционированного доступа. В этом случае операционная система работает в режиме *разделения времени*, т.е. обслуживает многих пользователей, работающих каждый со своего терминала.

Суть режима разделения времени состоит в следующем. Каждой программе, находящейся в оперативной памяти и готовой к исполнению, выделяется для исполнения фиксированный, задаваемый в соответствии с приоритетом пользователя интервал времени (интервал мультиплексирования). Если программа не выполнена до конца за этот интервал, ее исполнение принудительно прерывается, и программа переводится в конец очереди. Из начала очереди извлекается следующая программа, которая исполняется в течение соответствующего интервала мультиплексирования, затем поступает в конец очереди и т.д. в соответствии с циклическим алгоритмом. Если интервал мультиплексирования достаточно мал (~200 мс), а средняя длина очереди готовых к исполнению программ невелика (~10), то очередной квант времени выделяется программе каждые 2 с. В этих условиях ни один из пользователей практически не ощущает задержек, так как они сравнимы со временем реакции человека.

Одной из разновидностей режима разделения времени *является фоновый режим*, когда программа с более низким *приоритетом* работает на фоне программы с более высоким приоритетом. Работа в фоновом режиме реального времени аналогична работе секретаря руководителя. Секретарь занимается текущими делами до тех пор, пока начальник не дал срочное поручение.

Помимо рассмотренных режимов организации вычислительного процесса, все большее распространение получает схема, при которой ЭВМ управляет некоторым внешним процессом, обрабатывая данные и информацию, непосредственно поступающую от объекта управления. Поскольку определяющим фактором являются реально поступающие от объекта управления данные, такой режим называют *режимом реального времени*, а его организация возлагается на специализированную операционную систему.

По реализации интерфейса пользователя различают *неграфические* и *графические операционные системы.* Неграфические операционные системы реализуют *интерфейс командной строки.* Основным устройством управления в данном случае является клавиатура. Управляющие команды вводят в поле командной строки, где их можно и редактировать. Исполнение команды начинается после ее утверждения, например нажатием клавиши ENTER. Для компьютеров платформы *IBM PC* интерфейс командной строки обеспечивается семейством операционных систем под общим названием *MS-DOS* (версии от *MS-DOS* 1.0 до *MS-DOS* 6.2).

Графические операционные системы реализуют более сложный тип интерфейса, в котором в качестве органа управления кроме клавиатуры может использоваться мышь или адекватное устройство позиционирования. Работа с графической операционной системой основана на взаимодействии активных и пассивных экранных элементов управления.

В качестве активного элемента управления выступает *указатель мыши* *–* графический объект, перемещение которого на экране синхронизировано с перемещением мыши.

В качестве пассивных элементов управления выступают графические *элементы управления приложений* (экранные кнопки, значки, переключатели, флажки, раскрывающиеся списки, строки меню и многие другие).

Характер взаимодействия между активными и пассивными элементами управления выбирает сам пользователь. В его распоряжении приемы наведения указателя мыши на элемент управления, щелчки кнопками мыши и другие средства.

Все операционные системы обеспечивают свой автоматический запуск. Для дисковых операционных систем в специальной (*системной*) области диска создается запись программного кода. Обращение к этому коду выполняют программы, находящиеся в базовой системе ввода-вывода (*BIOS*). Завершая свою работу, они дают команду на загрузку и исполнение содержимого системной области диска.

Все современные дисковые операционные системы обеспечивают создание файловой системы, предназначенной для хранения данных на дисках и обеспечения доступа к ним.

К функции обслуживания файловой структуры относятся следующие операции, происходящие под управлением операционной системы:

* создание файлов и присвоение им имен;
* создание каталогов (папок) и присвоение им имен;
* переименование файлов и каталогов (папок);
* копирование и перемещение файлов между дисками компьютера и между каталогами (папками) одного диска;
* удаление файлов и каталогов (папок);
* навигация по файловой структуре с целью доступа к заданному файлу, каталогу (папке);
* управление атрибутами файлов.

*Файл –* это именованная последовательность байтов произвольной длины. Поскольку из этого определения вытекает, что файл может иметь нулевую длину, то фактически создание файла состоит в присвоении ему имени и регистрации его в файловой системе – это одна из функций операционной системы. Даже когда мы создаем файл, работая в какой-то прикладной программе, в общем случае для этой операции привлекаются средства операционной системы.

По способам именования файлов различают “короткое” и “длинное” имя. До появления операционной системы Windows 95 общепринятым способом именования файлов на компьютерах *IBM PC* было *соглашение 8.3.* Согласно этому соглашению, принятому в *MS-DOS,* имя файла состоит из двух частей: собственно *имени* и *расширения имени.* На имя файла отводится 8 символов, а на его расширение – 3 символа. Имя от расширения отделяется точкой. Как имя, так и расширение могут включать только алфавитно-цифровые символы латинского алфавита.

*Соглашение 8.3* не является стандартом, и потому в ряде случаев отклонения от правильной формы записи допускаются как операционной системой, так и ее приложениями. Так, например, в большинстве случаев система “не возражает” против использования некоторых специальных символов (восклицательный знак, символ подчеркивания, дефис, тильда и т. п.), а некоторые версии *MS-DOS* даже допускают использование в именах файлов символов русского и других алфавитов. Сегодня имена файлов, записанные в соответствии с *соглашением 83,* считаются “короткими”.

Основным недостатком “коротких” имен является их низкая содержательность. Далеко не всегда удается выразить несколькими символами характеристику файла, поэтому с появлением операционной системы Windows 95 было введено понятие “длинного” имени. Такое имя может содержать до 256 символов. Этого вполне достаточно для создания содержательных имен файлов. “Длинное” имя может содержать любые символы, кроме девяти специальных: \ / : \* ? " < > |. В имени разрешается использовать пробелы и несколько точек. Расширением имени считаются все символы, идущие после последней точки.

**Вспомогательные программы (утилиты)**

К системным программам можно также отнести большое количество так называемых утилит, т.е. программ вспомогательного назначения. Часто утилиты объединяются в комплексы, наиболее популярны комплексы Norton Utilities, PC Tools Deluxe и Mace Utilities.

*Программы-упаковщики* позволяют за счет применения специальных методов «упаковки» информации сжимать информацию на дисках, т.е. создавать копии файлов меньшего размера, а также объединять копии нескольких файлов в один архивный файл. Применение программ-упаковщиков очень полезно при создании архива файлов, так как в большинстве случаев значительно удобнее хранить на дисках или кассетах для стримера файлы, предварительно сжатые программами-упаковщиками. Наиболее популярны упаковщики PKZIP/PKUNZIP, ARJ и RAR. Следует заметить, что различные упаковщики не совместимы друг с другом – архивный файл, созданный одним упаковщиком, чаще всего нельзя прочесть другим.

*Программы для создания резервных копий информации на дисках* позволяют быстро скопировать информацию, находящуюся на жестком диске компьютера, на диски или кассеты стримера. Из этих программ широко используются Norton Backup (для DOS и для Windows), FastBack Plus. Соответствующие функции имеются в комплексе PC Tools Deluxe.

*Антивирусные программы* предназначены для предотвращения заражения компьютерным вирусом и ликвидации последствий заражения вирусом.

*Коммуникационные программы* предназначены для организации обмена информацией между компьютерами. Программы типа Brooklin Bridge, DeskLink, LapLink или FastLynx позволяют удобно пересылать файлы с одного компьютера на другой при соединении кабелем их последовательных портов (некоторые программы – при соединении параллельных портов, что обеспечивает большую скорость). Аналогичные возможности имеются в пакетах Norton Commander и PC Tools Deluxe.

Другой вид программ (Telemate, Procomm, DataLine и т.д.) обеспе­чивает возможность связи компьютеров по телефонной сети (при наличии модема). Программы типа WinFax Pro, BitFax, Fax-It и другие дают возможность посылать и принимать телефаксные сообщения при наличии в компьютере встроенной платы факс-модема.

*Программы для диагностики компьютера* позволяют проверить конфигурацию компьютера (количество памяти, ее использование, типы дисков и т.д.), а также проверить работоспособность устройств компьютера (прежде всего жестких дисков). Большой популярностью пользуются программы Check-It, NDiags, Control Room и System Sleuth. Для проверки работоспособности дисков весьма полезны программы Disk Technician Advanced, Calibrate из комплекса Norton Utilities и др. Они позволяют выявить «намечающиеся» дефекты дисков (возникающие из-за износа магнитной поверхности диска) и предотвратить потерю данных, хранящихся на диске.

*Программы-кэши для диска* убыстряют доступ к информации на диске путем организации в оперативной памяти кэш-буфера, содержащего наиболее часто используемые участки диска. Чаще всего для кэша используется дополнительная или расширенная память компьютера. Некоторые контроллеры дисков имеют в своем составе встроенный кэш-буфер, но обычно их производительность не намного выше, а стоимость значительно больше, чем у программы-кэша и соответствующего количества дополнительной памяти. Весьма хорошую производительность показывают программы SmartDrv, NCache и Super PC-Kwik.

*Программы для оптимизации дисков* позволяют обеспечить более быстрый доступ к информации на диске за счет оптимизации размещения данных на диске. Эти программы перемещают все участки каждого файла друг к другу (устраняют фрагментацию), собирают все файлы в начале диска и т.д., за счет чего уменьшается число перемещений головок диска (т.е. ускоряется доступ к данным) и снижается износ диска. Из программ для оптимизации дисков широко используются SpeeDisk из комплекса Norton Utilities, FdstTrax и др.

*Программы динамического сжатия дисков* (например, Stacker, DoubleSpace, SuperStor и др.) позволяют увеличить количество информации, хранимой на дисках путем ее динамического сжатия. Эти программы сжимают информацию при записи на диск, а при чтении восстанавливают ее в исходном виде. Таким образом, для пользователя эти программы незаметны (как говорят, «прозрачны»), они проявляются только увеличением емкости дисков и изменением скорости доступа (кстати, скорость доступа при этом может не уменьшиться, а даже увеличиться). Если на диске хранятся программы, то увеличение емкости невелико – в 1,5 раза, но для баз данных оно может достигать 4-5 раз.

*Программы для управления памятью* обеспечивают более гибкое использование оперативной памяти компьютера. Программы типа Software Carousel и Switch-It дают возможность загрузить в память компьютера несколько программ и «переключаться» с одной на другую с помощью нескольких нажатий клавиш. Программы Mark/Release, PopDrop Plus и другие обеспечивают эффективное управление резидентными программами, в частности «выгрузку» их из памяти после того, как в них отпадает необходимость.

*Программы для печати экрана* бывают весьма полезны при использовании графических программ для вывода на печать содержимого.

**Прикладное программное обеспечение**

Прикладные программы предназначены для того, чтобы обеспечить применении вычислительной техники в различных сферах деятельности человека. Помимо создания новых программных продуктов разработчики прикладных программ большие усилия тратят на совершенствование и модернизацию популярных систем создание их новых версий. Новые версии, как правило, поддерживают старые сохраняя преемственность, и включают в себя базовый минимум (стандарт) возможностей.

Прикладное программное обеспечение (ППО) делится на : ППО общего назначения, ППО специального назначения, ППО профессионального уровня.

Несмотря на широкие возможности использования компьютеров для обработки самой разной информации, самыми популярными являются программы, предназначенные для работы с текстами – текстовые редакторы и издательские системы. Текстовыми редакторами называют программы для ввода, обработки, хранения и печатания текстовой информации в удобном для пользователя виде. Эксперты оценивают использование компьютера в качестве печатающей машинки в 80%.

Большую популярность приобрели программы обработки графической информации. Компьютерная графика в настоящее время является одной из самых динамично развивающихся областей программного обеспечения. Она включает в себя ввод, обработку и вывод графической информации – чертежей, рисунков, картин, текстов и т.д. – средствами компьютерной техники. Различные типы *графических систем* позволяют быстро строить изображения, вводить иллюстрации с помощью сканера или видеокамеры, создавать анимационные ролики.

Графические редакторы позволяют пользоваться различным инструментарием художника, стандартными библиотеками изображений, наборами стандартных шрифтов, редактированием изображений, копированием и перемещением фрагментов по страницам экрана и др. Для выполнения расчетов и дальнейшей обработки числовой информации существуют специальные программы – электронные таблицы. В процессе деятельности любого специалиста часто требуется представить результаты работы в виде таблиц, где одна часть полей занята исходными данными, а другая – результатами вычислений и графического анализа. Характерными для них является большой объем перерабатываемой информации, необходимость многократных расчетов при изменении исходных данных. Автоматизацией подобной рутинной работы и занимаются электронные таблицы.

Одним из наиболее перспективных направлений развития вычислительной техники является создание специальных аппаратных средств для хранения гигантских массивов информационных данных, и последующей нечисловой обработки их – поиска и сортировки. Для компьютерной обработки подобных баз данных используют *системы управления базами данных*. СУБД – это набор средств программного обеспечения, необходимых для создания, обработки и вывода записей баз данных. Различают несколько типов СУБД: *иерархические*, *сетевые*, *реляционные.* При работе с СУБД выделяют несколько последовательных этапов:

* проектирование базы данных;
* создание структуры базы данных;
* заполнение базы данных;
* просмотр и редактирование базы данных;
* сортировку базы данных;
* поиск необходимой записи;
* выборку информации;
* создание отчетов.

Как правило, большинство популярных систем управления базами данных поддерживают эти этапы и предоставляют удобный инструментарий для их реализации.

Желание объединить функции различных прикладных программ в единую систему привело к созданию интегрированных систем. Универсальные *интегрированные системы* разрабатывались по принципу единой системы, содержащей в качестве элементов текстовые и графические редакторы, электронные таблицы и системы управления базами данных. Примеры: Framework, Works, Мастер. Современная концепция интеграции программных средств – кооперация отдельных прикладных программных систем по типу широко известного пакета Microsoft Office. Сами системы входящие в пакет, являются независимыми, более того, они сами представляют локально интегрированный пакет, поскольку помимо основной своей задачи, поддерживают функции других систем. Например, текстовый редактор Word обладает возможностью манипулировать с электронными таблицами и базами, а в электронной таблице Excel встроен мощный текстовый редактор. Для сопряжения информационных данных из различных программных систем в них предусматривают импорт-экспортную систему обмена с перекодировкой форматов представления данных.

Разработчики создают *специальные* программные системы целевого назначения для специалистов в некоторой предметной области. Такие программы называют авторскими инструментальными системами. Авторская система представляет интегрированную среду с заданной интерфейсной оболочкой, которую пользователь может наполнить информационным содержанием своей предметной области

*Экспертная система* – это программа, которая ведет себя подобно эксперту в некоторой узкой прикладной области. Экспертные системы призваны решать задачи с неопределенностью и неполными исходными данными, требующие для своего решения экспертных знаний.

Кроме того эти системы должны уметь объяснять свое поведение и свое решение.

Принципиальным отличием экспертных систем от других программ является их адаптивность, т.е. изменчивость в процессе самообучения.

Принято выделять в экспертных системах три основных модуля:

– модуль базы знаний;

– модуль логического вывода;

– интерфейс с пользователем.

Экспертные системы, являющиеся основой искусственного интеллекта, получили широкое распространение в науке (классификация животных и растении по видам, химический анализ), в медицине (постановка диагноза, анализ электрокардиограмм, определение методов лечения), в технике (поиск неисправностей в технических устройствах, слежение за полетом космических кораблей и спутников), в политологии и социологии, криминалистике, лингвистике и т.д.

В последнее время широкую популярность получили программы обработки гипертекстовой информации. Гипертекст – это форма организации текстового материала не в линейной последовательности, а в форме указаний возможных переходов (ссылок), связей между отдельными его фрагментами. В обычном тексте используется обычный линейный принцип размещения информации и доступ к нему осуществляется последовательно. В *гипертекстовых системах* информация напоминает текст энциклопедии, и доступ к любому выделенному фрагменту текста осуществляется произвольно по ссылке. Организация информации в гипертекстовой форме используется при создании справочных пособий, словарей, контекстной помощи (Help) в прикладных программах.

Расширение концепции гипертекста на графическую и звуковую информацию приводит к понятию гипермедиа. Идеи *гипермедиа* получили распространение в сетевых технологиях, в частности в Интернет-технологиях. Технология WWW позволила структурировать громадные мировые информационные ресурсы посредством гипертекстовых ссылок. Появились программные средства, позволяющие создавать подобные Web-странички. Стали развиваться механизмы поиска нужной информации в лабиринте информационных потоков. Популярными поисковыми средствами в Интернет являются Yahoo, AltaVista, Magellan, Rambler и др.

*Мультимедиа* – это взаимодействие визуальных и аудиоэффектов под управлением интерактивного программного обеспечения. Появление и широкое распространение компакт-дисков сделало эффективным использование мультимедиа в рекламной и информационной службе, сетевых телекоммуникационных технологиях, обучении.

Мультимедийные игровые и обучающие системы начинают вытеснять традиционные «бумажные библиотеки». Сегодня в библиотеках DVD-ROM можно «гулять» по музеям, Московскому Кремлю и т.д. с помощью «электронного путеводителя».

Каждая прикладная программа группы *профессионального* прикладного программного обеспечения ориентируются на достаточно узкую предметную область, но проникает в нее максимально глубоко. Так функционируют **АСНИ** – автоматизированные системы научных исследований, каждая из которых «привязана» к определенной области науки, **САПР** **–** системы автоматизированного проектирования, каждая из которых также работает в узкой области, **АСУ** – автоматизированные системы управления.

Наконец, еще раз подчеркнем не только условность предложенной выше классификации, но и наличие пересечений. Так, каждую конкретную экспертную систему вполне можно отнести к ППО профессионального уровня; принцип гипертекста реализован в ряде авторских систем и т.д.

**БАЗЫ ДАННЫХ**

**Базы данных и системы управления базами данных**

*База данных* – это организованная структура, предназначенная для хранения информации. Сегодня большинство *систем управления базами данных* (*СУБД*) позволяют размещать в своих структурах не только данные, но и методы (то есть программный код), с помощью которых происходит взаимодействие с потребителем или с другими программно-аппаратными комплексами. *Таким образом, в современных базах данных хранятся отнюдь не только данные, но и информация.*

Рассмотрим базу данных крупного банка. В ней есть все необходимые сведения о клиентах, об их адресах, кредитной истории, состоянии расчетных счетов, финансовых операциях и т.д. Доступ к этой базе имеется у достаточно большого количества сотрудников банка, но среди них вряд ли найдется такое лицо, которое имеет доступ ко *всей базе полностью* и при этом способно единолично вносить в нее произвольные изменения. Кроме данных, база содержит *методы и средства,* позволяющие каждому из сотрудников оперировать только с теми данными, которые входят в его компетенцию. В результате взаимодействия данных, содержащихся в базе, с методами, доступными конкретным сотрудникам, образуется информация, которую они потребляют и на основании которой в пределах собственной компетенции производят ввод и редактирование данных.

С понятием *базы данных* тесно связано понятие *системы управления базой данных.* Это комплекс программных средств, предназначенных для создания структуры новой базы, наполнения ее содержимым, редактирования содержимого и визуализации информации. Под *визуализацией информации базы* понимается отбор отображаемых данных в соответствии с заданным критерием, их упорядочение, оформление и последующая выдача на устройство вывода или передача по каналам связи.

В мире существует множество систем управления базами данных. Несмотря на то что они могут по-разному работать с разными объектами и предоставляют пользователю различные функции и средства, большинство *СУБД* опираются на единый устоявшийся комплекс основных понятий. Это дает возможность рассмотреть одну систему и обобщить ее понятия, приемы и методы на весь класс *СУБД.* В качестве такого учебного объекта рассмотрим *СУБД* Microsoft Access, входящую в пакет Microsoft Office.

**Структура простейшей базы данных**

Если в базе нет никаких данных (*пустая база*), то это все равно полноценная база данных. Хотя данных в базе и нет, но информация в ней все-таки есть – это *структура базы.* Она определяет методы занесения данных и хранения их в базе. Простейший “некомпьютерный” вариант базы данных – деловой ежедневник, в котором каждому календарному дню выделено по странице. Даже если в нем не записано ни строки, он не перестает быть ежедневником, поскольку имеет структуру, четко отличающую его от записных книжек, рабочих тетрадей и прочей писчебумажной продукции.

Основными объектами любой базы данных являются ее таблицы. Простейшая база данных имеет хотя бы одну таблицу. Соответственно, структура простейшей базы данных тождественно равна структуре ее таблицы. Структуру двумерной таблицы образуют столбцы и строки. Их аналогами в структуре простейшей базы данных являются *поля* и *записи* (рис. 13)*.* Если записей в таблице пока нет, значит, ее структура образована только набором полей. Изменив состав полей базовой таблицы (или их свойства), мы изменяем структуру базы данных и, соответственно, получаем новую базу данных.



Рис. 13. Простейшая таблица базы данных

**Свойства полей базы данных**

Поля базы данных не просто определяют структуру базы – они еще определяют групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей. Основные свойства полей таблиц баз данных на примере СУБД Microsoft Access.

* Имя поля – определяет, как следует обращаться к данным этого поля при автоматических операциях с базой (по умолчанию имена полей используются в качестве заголовков столбцов таблиц).
* Тип поля – определяет тип данных, которые могут содержаться в данном поле.
* Размер поля – определяет предельную длину (в символах) данных, которые могут размещаться в данном поле.
* Формат поля – определяет способ форматирования данных в ячейках, принадлежащих полю.
* Маска ввода – определяет форму, в которой вводятся данные в поле (средство автоматизации ввода данных).
* Подпись – определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство Имя поля).
* Значение по умолчанию – то значение, которое вводится в ячейки поля автоматически (средство автоматизации ввода данных).
* Условие на значение – ограничение, используемое для проверки правильности ввода данных (средство автоматизации ввода, которое используется, как правило, для данных, имеющих числовой тип, денежный тип или тип даты).
* Сообщение об ошибке – текстовое сообщение, которое выдается автоматически при попытке ввода в поле ошибочных данных (проверка ошибочности выполняется автоматически, если задано свойство Условие на значение).
* Обязательное поле – свойство, определяющее обязательность заполнения данного поля при наполнении базы;
* Пустые строки – свойство, разрешающее ввод пустых строковых данных (от свойства Обязательное поле отличается тем, что относится не ко всем типам данных, а лишь к некоторым, например к текстовым).
* Индексированное поле – если поле обладает этим свойством, все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. Кроме того, для индексированных полей можно сделать так, что значения в записях будут проверяться по этому полю на наличие повторов, что позволяет автоматически исключить дублирование данных.

**Типы данных**

К основными типами данных относят: текст, числа и формулы. Таблицы баз данных, как правило, допускают работу с гораздо большим количеством разных типов данных. Так, например, базы данных Microsoft Access работают со следующими типами данных (рис. 14).



Рис. 14. Таблица с полями некоторых типов

* Текстовый – тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов).
* Поле Мемо – специальный тип данных для хранения больших объемов текста (до 65 535 символов). Физически текст не хранится в поле. Он хранится в другом месте базы данных, а в поле хранится указатель на него, но для пользователя такое разделение заметно не всегда.
* Числовой – тип данных для хранения действительных чисел.
* Дата/время – тип данных для хранения календарных дат и текущего времени.
* Денежный – тип данных для хранения денежных сумм. Теоретически, для их записи можно было бы пользоваться и полями числового типа, но для денежных сумм есть некоторые особенности (например, связанные с правилами округления), которые делают более удобным использование специального типа данных, а не настройку числового типа.
* Счетчик – специальный тип данных для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием. Естественное использование – для порядковой нумерации записей.
* Логический – тип для хранения логических данных (могут принимать только два значения, например Да или Нет).
* Поле объекта OLE – специальный тип данных, предназначенный для хранения объектов *OLE,* например мультимедийных. Реально, конечно, такие объекты в таблице не хранятся. Как и в случае полей MEMO, они хранятся в другом месте внутренней структуры файла базы данных, а в таблице хранятся только указатели на них (иначе работа с таблицами была бы чрезвычайно замедленной).
* Гиперссылка – специальное поле для хранения адресов *URL* Web-объектов Интернета. При щелчке на ссылке автоматически происходит запуск браузера и воспроизведение объекта в его окне.
* Мастер подстановок – это не специальный тип данных. Это объект, настройкой которого можно автоматизировать ввод в данных поле так, чтобы не вводить их вручную, а выбирать из раскрывающегося списка.

**Безопасность баз данных**

Базы данных – это тоже файлы, но работа с ними отличается от работы с файлами других типов, создаваемых прочими приложениями. Для баз данных предъявляются особые требования с точки зрения безопасности, поэтому в них реализован другой подход к сохранению данных.

Базы данных – это особые структуры. Информация, которая в них содержится, очень часто имеет общественную ценность. Нередко с одной и той же базой (например, с базой регистрации автомобилей в ГИБДД) работают тысячи людей по всей стране. От информации, которая содержится в некоторых базах, может зависеть благополучие множества людей. Поэтому целостность содержимого базы не может и не должна зависеть ни от конкретных действий некоего пользователя, забывшего сохранить файл перед выключением компьютера, ни от перебоев в электросети.

Проблема безопасности баз данных решается тем, что в СУБД для сохранения информации используется двойной подход. В части операций, как обычно, участвует операционная система компьютера, но некоторые операции сохранения происходят в обход операционной системы.

Операции изменения структуры базы данных, создания новых таблиц или иных объектов происходят при сохранении файла базы данных. Об этих операциях СУБД предупреждает пользователя. Это, так сказать, глобальные операции. Их никогда не проводят с базой данных, находящейся в коммерческой эксплуатации, – только с ее копией. В этом случае любые сбои в работе вычислительных систем не страшны.

С другой стороны, операции по изменению содержания данных, не затрагивающие структуру базы, максимально автоматизированы и выполняются без предупреждения. Если работая с таблицей данных, мы что-то в ней меняем в составе данных, то изменения сохраняются немедленно и автоматически.

Обычно, решив отказаться от изменений в документе, его просто закрывают без сохранения и вновь открывают предыдущую копию. Этот прием работает почти во всех приложениях, но только не в СУБД. Все изменения, вносимые в таблицы базы, сохраняются на диске без нашего ведома, поэтому попытка закрыть базу “без сохранения” ничего не даст, так как все уже сохранено. Таким образом, редактируя таблицы баз данных, создавая новые записи и удаляя старые, мы как бы работаем с жестким диском напрямую, минуя операционную систему.

**Проектирование баз данных**

Обычно с базами данных работают две категории исполнителей. Первая категория – *проектировщики.* Их задача состоит в разработке структуры таблиц базы данных и согласовании ее с заказчиком. Кроме таблиц проектировщики разрабатывают и другие объекты базы данных, предназначенные, с одной стороны, для автоматизации работы с базой, а с другой стороны – для ограничения функциональных возможностей работы с базой (если это необходимо из соображений безопасности). Проектировщики не наполняют базу конкретными данными (заказчик может считать их конфиденциальными и не предоставлять посторонним лицам). Исключение составляет экспериментальное наполнение модельными данными на этапе отладки объектов базы.

Вторая категория исполнителей, работающих с базами данных, – *пользователи.* Они получают исходную базу данных от проектировщиков и занимаются ее наполнением и обслуживанием. В общем случае пользователи не имеют средств доступа к управлению структурой базы – только к данным, да и то не ко всем, а к тем, работа с которыми предусмотрена на конкретном рабочем месте.

Соответственно, система управления базами данных имеет два режима работы: *проектировочный* и *пользовательский.* Первый режим предназначен для создания или изменения структуры базы и создания ее объектов. Во втором режиме происходит использование ранее подготовленных объектов для наполнения базы или получения данных из нее.

***Запросы*** служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С помощью запросов выполняют такие операции как отбор данных, их сортировку и фильтрацию. С помощью запросов можно выполнять преобразование данных по заданному алгоритму, создавать новые таблицы, выполнять автоматическое наполнение таблиц данными, импортированными из других источников, выполнять простейшие вычисления в таблицах и многое другое.

Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе *временную результирующую таблицу.* Если хотят подчеркнуть факт “временности” этой таблицы, то ее еще называют *моментальным снимком.* Когда мы работаем с основными таблицами базы, мы физически имеем дело с жестким диском, то есть с очень медленным устройством (напомним, что это связано с особенностью сохранения данных, описанной выше). Когда же на основании запроса мы получаем результирующую таблицу, то имеем дело с электронной таблицей, не имеющей аналога на жестком диске, – это только *образ* отобранных полей и записей. Разумеется, работа с “образом” происходит гораздо быстрее и эффективнее – это еще одно основание для того, чтобы широко использовать запросы.

Недостатком упорядоченных табличных структур является сложность их обновления, поскольку при внесении новых записей нарушается упорядоченность – приходится переделывать всю таблицу. В системах управления базами данных и эта проблема решается благодаря запросам. Основной принцип состоит в том, что от базовых таблиц никакой упорядоченности не требуется. Все записи в основные таблицы вносятся только в *естественном* порядке по мере их поступления, то есть в неупорядоченном виде. Если же пользователю надо видеть данные, отсортированные или отфильтрованные по тому или иному принципу, он просто использует соответствующий запрос (рис. 15). Если нужного запроса нет, он обращается к проектировщику и просит его такой запрос сделать и предоставить.



Рис. 15. Два запроса, сформированные на основе одной таблицы

***Формы.*** Если запросы – это специальные средства для отбора и анализа данных, то формы – это средства для ввода данных. Смысл их тот же – предоставить пользователю средства для заполнения только тех полей, которые ему заполнять положено. Одновременно с этим в форме можно разместить специальные элементы управления (счетчики, раскрывающиеся списки, переключатели, флажки и прочие) для автоматизации ввода. Преимущества форм раскрываются особенно наглядно, когда происходит ввод данных с заполненных бланков. В этом случае форму делают графическими средствами так, чтобы она повторяла оформление бланка – это заметно упрощает работу наборщика, снижает его утомление и предотвращает появление печатных ошибок. На сопроводительном рисунке приведен пример простейшей формы для ввода данных.

С помощью форм данные можно не только вводить, но и отображать. Запросы тоже отображают данные, но делают это в виде результирующей таблицы, не имеющей почти никаких средств оформления. При выводе данных с помощью форм можно применять специальные средства оформления (рис. 16).



Рис. 16. Форма для оформленного вывода данных

***Отчеты.*** По своим свойствам и структуре отчеты во многом похожи на формы, но предназначены только для вывода данных, причем для вывода не на экран, а на печатающее устройство (принтер). В связи с этим отчеты отличаются тем, что в них приняты специальные меры для группирования выводимых данных и для вывода специальных элементов оформления, характерных для печатных документов (верхний и нижний колонтитулы, номера страниц, служебная информация о времени создания отчета и т. п.) (рис. 17).



Рис. 17. Пример простейшего отчета

**Разработка структуры базы данных**

Выяснив основную часть данных, которые заказчик потребляет или поставляет, можно приступать к созданию структуры базы, то есть структуры ее основных таблиц.

1. Работа начинается с составления генерального списка полей – он может насчитывать десятки и даже сотни позиций.
2. В соответствии с типом данных, размещаемых в каждом поле, определяют наиболее подходящий тип для каждого поля.
3. Далее распределяют поля генерального списка по базовым таблицам. На первом этапе распределение производят по функциональному признаку. Цель – обеспечить, чтобы ввод данных в одну таблицу производился, по возможности, в рамках одного подразделения, а еще лучше – на одном рабочем месте.

Наметив столько таблиц, сколько подразделений охватывает база данных, приступают к дальнейшему делению таблиц. Критерием необходимости деления является факт множественного повтора данных в соседних записях. На рис. 18 показана таблица, у которой в поле Адрес наблюдается повтор данных. Это явное свидетельство того, что таблицу надо поделить на две взаимосвязанные таблицы.



Рис. 18. Если данные в поле начинают повторяться, это признак того,

что таблицу стоит поделить

1. В каждой из таблиц намечают *ключевое поле.* В качестве такового выбирают поле, данные в котором повторяться не могут. Например, для таблицы данных о студентах таким полем может служить индивидуальный шифр студента. Для таблицы, в которой содержатся расписания занятий, такого поля можно и не найти, но его можно создать искусственным комбинированием полей “Время занятия” и “Номер аудитории”. Эта комбинация неповторима, так как в одной аудитории в одно и то же время не принято проводить два различных занятия.
2. С помощью карандаша и бумаги расчерчивают связи между таблицами. На рис. 19 показан пример взаимосвязи между группой таблиц, составляющих одну базу данных. Такой чертеж называется *схемой данных*.
3. Существует несколько типов возможных связей между таблицами. Наиболее распространенными являются связи “один ко многим” и “один к одному”. Связь между таблицами организуется на основе общего поля, причем в одной из таблиц оно обязательно должно быть ключевым, то есть на стороне “один” должно выступать ключевое поле, содержащее уникальные, неповторяющиеся значения. Значения на стороне “многие” могут повторяться.
4. Рассмотрим таблицу Клиенты (рис. 19). Здесь поле Код клиента является ключевым. Это понятно, поскольку у каждого клиента должен быть свой уникальный код, идентифицирующий его однозначно. Если мы рассмотрим таблицу Заказы, то увидим, что в ней код клиента не может быть уникальным, поскольку каждый клиент мог сделать сколь угодно много заказов. На схеме данных эти поля соединены *линией связи.* С одной стороны эта линия маркирована знаком “1”, с другой стороны – значком “бесконечность”. Это графический метод изображения связи “один ко многим”.



Рис. 19. Схема связей между таблицами

Ключевым полем в таблице заказов является Код заказа – он однозначно идентифицирует, кто, когда, что заказал и на какую сумму. Здесь же можно узнать, какой сотрудник принял заказ к исполнению. Поскольку один сотрудник может принять множество заказов, поле Код сотрудника в таблице заказов не является ни уникальным, ни ключевым, зато в таблице Сотрудники это поле уникально.

Про подобные таблицы говорят, что они *связаны реляционными отношениями.* Соответственно, системы управления, способные работать со связанными таблицами, называют *системами управления реляционными базами данных,* а схему данных в технической литературе могут называть *схемой реляционных отношений.*

**СУБД Microsoft Access 2000**

СУБД Microsoft Access 2000 предоставляет несколько средств создания каждого из основных объектов базы. Эти средства можно классифицировать как:

* ручные (разработка объектов в режиме Конструктора);
* автоматизированные (разработка с помощью программ-мастеров);
* автоматические – средства ускоренной разработки простейших объектов.

Соотношения между этими средствами понятны: ручные средства являются наиболее трудоемкими, но обеспечивают максимальную гибкость; автоматизированные и автоматические средства являются наиболее производительными, но и наименее гибкими

При разработке учебных таблиц и запросов рекомендуется использовать ручные средства – работать в режиме Конструктора. Использование мастеров ускоряет работу, но не способствует освоению понятий и методов.

При разработке учебных форм, отчетов и страниц доступа наоборот лучше пользоваться автоматизированными средствами, предоставляемыми мастерами. Это связано с тем, что для данных объектов большую роль играет внешний вид. Дизайн этих объектов весьма трудоемок, поэтому его лучше поручить программе, а учащемуся сосредоточиться на содержательной части работы.

Разработку макросов и модулей в данном пособии мы не рассматриваем. Эти средства ориентированы на профессиональных разработчиков баз данных, поэтому в рамках общетехнического курса “Информатики” для них недостаточно места.

**СИСТЕМЫ ПРОГРАММИРОВАНИЯ**

В самом общем случае для создания программы на выбранном языке программирования нужно иметь следующие компоненты.

1. *Текстовый редактор.* Так как текст программы записывается с помощью ключевых слов, обычно происходящих от слов английского языка, и набора стандартных символов для записи всевозможных операций, то формировать этот текст можно в любом редакторе, получая в итоге текстовый файл с *исходным текстом* программы. Лучше использовать специализированные редакторы, которые ориентированы на конкретный язык программирования и позволяют в процессе ввода текста выделять ключевые слова и идентификаторы разными цветами и шрифтами. Подобные редакторы созданы для всех популярных языков и дополнительно могут автоматически проверять правильность синтаксиса программы непосредственно во время ее ввода.
2. Исходный текст с помощью *программы-компилятора* переводится в машинный код. Если обнаружены синтаксические ошибки, то результирующий код создан не будет. На этом этапе уже возможно получение готовой программы, но чаще всего в ней не хватает некоторых компонентов, поэтому компилятор обычно выдает промежуточный *объектный код* (двоичный файл, стандартное расширение .OBJ).
3. Исходный текст большой программы состоит, как правило, из нескольких *модулей* (файлов с исходными текстами), потому что хранить все тексты в одном файле неудобно – в них сложно ориентироваться. Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем надо объединить в одно целое. Кроме того, к ним надо добавить машинный код подпрограмм, реализующих различные стандартные функции (например вычисляющих математические функции sin или In). Такие функции содержатся в *библиотеках* (файлах со стандартным расширением .LIB), которые поставляются вместе с компилятором. Сгенерированный код модулей и подключенные к нему стандартные функции надо не просто объединить в одно целое, а выполнить такое объединение с учетом требований операционной системы, то есть получить на выходе программу, отвечающую определенному формату.

Объектный код обрабатывается специальной программой *– редактором связей* или *сборищном,* который выполняет связывание объектных модулей и машинного кода стандартных функций, находя их в библиотеках, и формирует на выходе работоспособное приложение – *исполнимый код* для конкретной платформы.

Если по каким-то причинам один из объектных модулей или нужная библиотека не обнаружены (например, неправильно указан каталог с библиотекой), то сборщик сообщает об ошибке и готовой программы не получается.

1. Исполнимый код – это законченная программа, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .ЕХЕ или .СОМ.

Итак, для создания программы нужны:

* текстовый редактор;
* компилятор;
* редактор связей;
* библиотеки функций.

Как правило, в стандартную поставку входят как минимум три последних компонента, но хорошая *интегрированная система* включает в себя и специализированный текстовый редактор, причем почти все этапы создания программы в ней автоматизированы: после того как исходный текст введен, его компиляция и сборка выполняются одним нажатием клавиши. Это очень удобно, так как не требует ручной настройки множества параметров запуска компилятора и редактора связей, указывания им нужных файлов вручную и т. д. Процесс компиляции обычно демонстрируется на экране: показывается, сколько строк исходного текста откомпилировано, или выдаются сообщения о найденных ошибках.

В современных интегрированных системах имеется еще один компонент – *отладчик,* который позволяет анализировать работу программы во время ее выполнения. С его помощью можно последовательно выполнять отдельные операторы исходного текста *по шагам,* наблюдая при этом, как меняются значения различных переменных. Без отладчика разработать крупное приложение очень сложно.

В последние несколько лет в программировании наметился так называемый *визуальный подход.* До этого серьезным препятствием для разработки графических приложений была сложность создания различных элементов управления и контроля за их работой. Достаточно взглянуть на окно любой Windows-программы. В нем имеется множество стандартных элементов управления (кнопки, пункты меню, списки, переключатели и т. д.). Очень трудоемко вручную описывать процесс создания этих элементов в соответствии с требованиями Windows, на глазок определять координаты, отслеживать их состояние с помощью специальных команд. Например, для простой программы, складывающей два числа, потребуется один оператор (одна строка исходного текста) для выполнения нужного вычисления и сотни строк кода для подготовки приложения к работе в Windows, создания кнопки и пары полей ввода.

Этот процесс автоматизирован в *средах быстрого проектирования* (Rapid Application Development, *RAD-среды).* Все необходимые элементы оформления и управления создаются и обслуживаются не путем ручного программирования, а с помощью готовых визуальных *компонентов*, которые с помощью мыши “перетаскиваются” в проектируемое окно. Их свойства и поведение затем настраиваются с помощью простых редакторов, визуально показывающих характеристики соответствующих элементов. При этом вспомогательный исходный текст программы, ответственный за создание и работу этих элементов, генерируется *RAD*-средой автоматически, что позволяет сосредоточиться только на логике решаемой задачи. В результате программирование во многом заменяется на проектирование – подобный подход называется еще *визуальным программированием.*

Из универсальных языков программирования сегодня наиболее популярны следующие:

– Бейсик (Basic) – для освоения требует начальной подготовки (общеобразовательная школа);

– Паскаль (Pascal) – требует специальной подготовки (школы с углубленным изучением предмета и общетехнические ВУЗы);

– Си++ (C++), Ява (Java) – требуют профессиональной подготовки (специализированные средние и высшие учебные заведения).

Для каждого из этих языков программирования сегодня имеется немало систем программирования, выпускаемых различными фирмами и ориентированных на различные модели ПК и операционные системы. Наиболее популярны следующие визуальные среды быстрого проектирования программ для Windows:

* Basic: Microsoft Visual Basic
* Pascal: Borland Delphi
* C++: Borland C++Bulider
* Java: Symantec Cafe

**ФАЙЛЫ И КАТАЛОГИ НА ДИСКАХ**

**Файлы и их имена**

 Информация на магнитных дисках хранится в файлах. Файл – это поименованная область на диске или другом машинном носителе. В файлах могут храниться тексты программ, документы, готовые к выполнению программы и т.д.

Часто файлы разделяют на две категории – текстовые и двоичные. Текстовые файлы предназначены для чтения человеком. Они состоят из строк символов, причем каждая строка оканчивается двумя специальными символами «возврат каретки» (CR) и «новая строка» (LF). При редактировании и просмотре текстовых файлов эти специальные символы, как правило, не видны. В текстовых файлах хранятся тексты программ, командных файлов DOS и т.д. Файлы, не являющиеся текстовыми, называются двоичными.

 Текстовый файл, содержащий только символы с кодами до 127 (т.е. не содержащий русских букв и псевдографических символов), называется ASCII-файлом.

 Каждый файл на диске имеет обозначение, которое состоит из двух частей: имени и расширения (часто имя и расширение вместе также называются именем, как правило, это не приводит к путанице). Расширение имени файла является необязательным. Оно, как правило, описывает содержание файла, поэтому использование расширения весьма удобно. Многие программы устанавливают расширение имени файла и по нему можно узнать, какая программа создала файл:

\*.com, \*.exe готовые к выполнению программы

\*.bat командные (Batch) файлы;

\*.pas программы на Паскале;

\*.for программы на Фортране;

\*.с программы на Си;

\*.asm программы на Ассемблере;

\*.bak копия файла, делаемая перед его изменением.

\*.doc, \*.docx документы Word

\*.xls документы Excel

\*.pps документы PowerPoint

 В имени и расширении имени файла прописные и строчные латин­ские буквы являются эквивалентными.

**Папки. Файловая система**

 Имена файлов регистрируются на магнитных дисках в каталогах (или папках). Каталог (папка) – это специальное место на диске, в котором хранятся имена файлов, сведения о размере файлов, времени их последнего обновления, атрибуты (свойства) файлов и т.д. Если в каталоге (папке) хранится имя файла, то говорят, что этот файл находится в данном каталоге (папке). На каждом магнитном диске может быть несколько каталогов (папок). В каждом каталоге (папке) может быть много файлов, но каждый файл всегда регистрируется только в одном каталоге (папке).

 *Подкаталоги и надкаталоги.* Все каталоги (кроме корневого) на самом деле являются файлами специального вида. Каждый каталог имеет имя, и он может быть зарегистрирован в другом каталоге. Если каталог X зарегистрирован в каталоге Y, то говорят, что X – подкаталог Y, a Y – надкаталог или родительский каталог для X.

 *Имена каталогов*. Требования к именам каталогов те же, что к именам файлов. Как правило, расширение имени для каталогов не используется.

 *Корневой каталог*. На каждом магнитном диске имеется один главный, или корневой, каталог. В нем регистрируются файлы и подкаталоги (каталоги 1-го уровня). В каталогах 1-го уровня регистрируются файлы и каталоги 2-го уровня и т.д. Получается иерархическая древообразная структура каталогов на магнитном диске (рис. 20).

**Путь к файлу**

 Когда используется файл не из текущего каталога, необходимо указать, в каком каталоге этот файл находится. Это делается с помощью указания пути к файлу.

 Путь – это последовательность из имен каталогов или символов «..» , разделенных символом «\». Этот путь задает маршрут от текущего каталога или от корневого каталога диска к тому каталогу, в котором находится нужный файл.



Рис. 3.9. Пример файловой системы на диске

 Если путь начинается с символа «\», то маршрут вычисляется от корневого каталога диска, иначе – от текущего каталога. Каждое имя каталога в пути соответствует входу в подкаталог с таким именем, «..» соответствует входу в надкаталог.

 Полное имя файла имеет следующий вид (скобками [ и ] обозначаются необязательные элементы):

[дисковод:] [путь\]имя-файла

т.е. состоит из пути к каталогу, в котором находится файл, и имени файла, разделенных символом «\», перед которыми может стоять обо­значение дисковода.

Если дисковод не указан, то подразумевается текущий дисковод. Если путь не указан, то подразумевается текущий каталог.

 Полное имя файла полностью специфицирует, с каким файлом Вы хотите работать.