## § 2. БАЗЫ ДАННЫХ В СТРУКТУРЕ ИНФОРМАЦИОННЫХ СИСТЕМ

### 2.1. ОСНОВНЫЕ ПОНЯТИЯ

Понятие «база данных» в отличие от понятия «банк информации» изначально связано с компьютерными системами, с историей и развитием.

Исторически понятие базы данных возникло как альтернатива файловой организации данных при хранении с помощью ЭВМ (на магнитных носителях). Такая организация данных была характерна для прикладного программного обеспечения на начальном этапе распространения вычислительной техники. Файловая организация предполагала хранение данных в виде совокупности файлов, ориентированных на использование какой-либо одной прикладной программы, предназначенной для решения некоторой специфической задачи. Такая неуниверсальность в организации информации привела к большой избыточности (дублированию) при хранении, противоречивости данных, хранящихся в различных системах.

Для файловой организации данных свойственна высокая зависимость данных **от** программ, так что информация о форме хранения данных скрыта в сочетании «файл - программа».

Понятие «база данных» возникло в результате стандартизации и унификации данных, универсально организованных и хранящихся с помощью ЭВМ с целью использования для многих приложений. При этом описание данных уже не скрыто в программах, а явным образом декларируется и хранится в самой базе. База данных может быть определена как структурная совокупность данных, поддерживаемых в актуальном состоянии (в соответствии объектам некоторой предметной области) и служащая для удовлетворения информационных потребностей многих пользователей. Базы данных устраняют избыточность и противоречивость данных. Для поддержания актуальности данных, хранящихся в базе, получения сводок по информационным запросам, перехода к данным и программам пользователей служат системы управления базами данных (СУБД), описанные в общих чертах в гл. 2.

Основой СУБД являются два языка - язык описания данных (ЯОД) и язык манипулирования данными (ЯМД). С помощью языка описания данных администратор базы данных и программисты описывают структуру и содержимое базы данных. Язык манипулирования данными является средством, которое применяется пользователями или прикладным программистом для выполнения операций над данными, хранящимися в базе: добавления новых данных, изменения или удаления устаревших, упорядочения данных по тем или иным признакам, поиска данных в соответствии с запросами.

Интеграция данных в базе подразумевает совместное использование данных для решения различных прикладных задач и устраняет дублирование данных. Однако согласованное понимание и использование данных требует централизованного управления, которое называется администрированием данных. Подчеркнем, что в данном случае идет речь не о собственно значениях данных, а обихсмысле и форме.

Смысл элементов данных, способы их использования, источники, различные правила и ограничения представляют собой метаданные. В формализованном виде метаданные могут содержаться в словарях данных, обслуживающих базы данных. Основными функциями словарей данных являются

• установление единообразного понимания данных пользователями БД;

• эффективное управление элементами данных при модификации описания данных в системе;

• уменьшение избыточности;

• устранение противоречивости;

• упрощение проектирования БД и ее сопровождения, расширения и т.д.

Словари данных используются конечными пользователями при работе с системой на языке запросов, прикладными программистами при написании программ, системными программистами в процессе развития системы.

Коллектив специалистов, обслуживающих большие базы данных, включает администратора, аналитиков, системных и прикладных программистов.

Администратор - это специалист, имеющий представление об информационных потребностях конечных пользователей и отвечающий за определение, загрузку, защиту и эффективность базы данных.

Аналитики, обладая знаниями закономерностей соответствующей предметной области, в контакте с конечными пользователями строят формальные (математические) модели для задач конечного пользователя, которые являются исходным представлением задачи для прикладного программиста.

Прикладные программисты на основе представления задачи, полученного аналитиками, разрабатывают прикладные программы для решения задач конечных пользователей.

Системные программисты обеспечивают работоспособность операционной системы, систем программирования и СУБД, разрабатывают сервисные программы.

Приведем перечень важнейших требований, которым должны удовлетворять современные базы данных:

• адекватность базы данных предметной области;

• интегрированность данных;

• независимость данных;

• минимальная избыточность хранимых данных;

• целостность базы данных;

• обеспечение защиты от несанкционированного доступа или случайного уничтожения данных;

• гибкость и адаптивность структуры базы данных;

•динамичность данных и способность к расширению;

• возможность поиска по многим ключам.

### 2.2. ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Рассмотрим вопрос о проектировании баз данных. К любой базе данных возможен подход на каждом из следующих трех уровней (рис. 6.6):

• на уровне представлений данных конечного пользователя или прикладного программиста (внешнего представления);

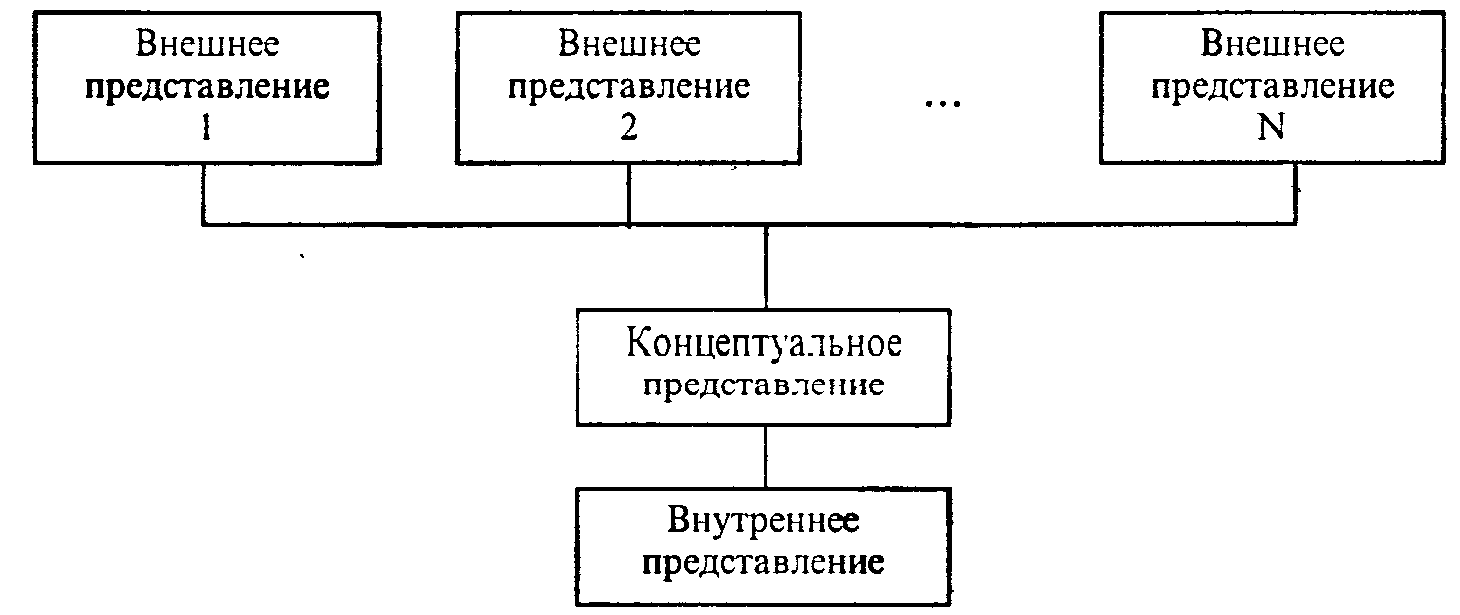
• на уровне концептуального представления данных (представления администратора);

• на уровне внутреннего представления данных (с позиции системного программиста) или представления реализации.

Под представлением данных понимаются правила организации и кодирования данных.

Представление данных конечного пользователя (внешнее представление) является совокупностью требований к данным со стороны некоторой конкретной задачи или программы.

Для конечного пользователя внешнее представление является совокупностью спецификаций и реальных форматов данных, отражающих конкретные информационные потребности при решении некоторой задачи (проблемы). Представление прикладного программиста отображает элементы данных и их взаимосвязи так, что из данных, хранящихся в базе, может извлекаться скрытая в них информация (например, возраст человека по дате рождения и текущей дате).



*Рис. 6.6.* Трехуровневое представление данных в концепции ANSI/SPARC '

Так как с данными обычно работают многие пользователи и прикладные программисты, имеется множество частично перекрывающихся внешних представлений данных.

Концептуальное представление данных является интегрированным определением данных на основе объединения внешних представлений данных для всей совокупности приложений, т.е. достаточно полной моделью предметной области. Структура данных на концептуальном уровне называется концептуальной схемой и описывает семантику данных.

Внутреннее (физическое) представление, или представление реализации, выражает представление данных системными программистами и связано с организацией хранения данных на физических носителях информации (запоминающих устройствах) и их обработкой. Основными понятиями внутреннего представления являются физические блоки, хранимые записи, указатели и т.д. Внутреннее представление обеспечивает доступ к данным на логическом уровне и скрывает от прикладных программистов и пользователей многие технические детали манипулирования данными и методов доступа кним.

Наибольший интерес для нас представляет концептуальное представление данных, связанное с развитой в 70-80-е годы теорией баз и банков данных и направленное на унификацию данных и уменьшениеизбыточности при интегрировании внешних представлений в концептуальное.

Концептуальное представление основано на определенной модели данных.Этоттермин, впервые введенный в 70-х годах основоположником теории баз данных Дж.Коддом, в современной трактовке отображает совокупность правил порождения структур данных в базах данных, последовательности их изменения. Различают три основные типа модели данных: иерархический, сетевой и реляционный, рассмотренные ранее в гл. 2.

Модель данных предопределяет множество выводимых допустимых типов данных и отношений между ними и является основой для построения модели конкретной базы данных.

Модель базы данных является средством интерпретации содержимого базы данных и реализации операции по обработке и управлению данными.

Проектирование баз данных представляет собой длительный, трудоемкий и слабоформализованный процесс, от которого зависит жизнеспособность и эффективность проектируемой базы данных, ее способность к развитию. Важную роль при проектировании базы данных играет методология построения концептуальных моделей предметной области, включающая методы и средства, позволяющие спроектировать базу данных, удовлетворяющую заданным целям и требованиям пользователей и прикладных программистов.

Такими средствами моделирования являются системный анализ, методы экспертных оценок, с помощью которых в концептуальной модели совмещаются концептуальное представление объективно существующей предметной области и концептуальное представление субъективных информационных требований к данным со стороны пользователей и прикладных программистов.

В недавнем прошлом процесс проектирования баз данных был ориентирован, в основном, на требования пользователей и прикладных программистов (ПП-информация) и учитывал текущие или предвидимые приложения. В этом случае база данных создавалась сравнительно легко и быстро. Однако такие базы данных оказывались неприспособленными к обработке неформализованных, изменяющихся, не предвиденных ранее запросов и приложений, не имели стимулов к дальнейшему развитию. Поэтому важную роль при проектировании стала играть информация о предметной области (ПО-информация), не зависящая напрямую от существующих приложений и обеспечивающая гибкость, адаптивность и универсальность данных, пригодность всей системы к развитию и использованию для незапланированных будущих приложений.

Современная методология проектирования баз данных и построения концептуальных моделей основывается на одновременном учете ПО- и ПП-информаций. ПО-информация в этом случае используется для построения первоначальной информационной структуры данных, а ПП-информация - для совершенствования последней с целью повышения эффективности обработки данных.

Процесс построения концептуальной модели разделяется на следующие этапы:

1) сбор и содержательный анализ априорной информации о предметной области и прикладных задачах пользователей;

2) концептуальный анализ данных и синтез концептуальной модели.

На этапе сбора данных проводятся наблюдения и измерения, собираются отчеты и различные документы, интервьюируются специалисты в данной предметной области, выявляется перечень задач организации и ее структурных подразделений. Сбор информации начинается с определения сферы применения базы данных. Сфера применения базы данных должна определяться независимо от прикладных задач и охватывать все функциональные подразделения организации. Для этого проводятся собеседования с руководством организации с целью выявления отделов данной организации и внешних организаций, связанных с информационным обеспечением текущих и планируемых прикладных задач, а также возможных в будущем изменений в деятельности организации. Далее проводятся дополнительные собеседования в подразделениях организации с целью выявления совместно используемых данных. На этом же этапе собранные данные анализируются на предмет устранения дублирования и противоречивости данных, неоднозначности их определений и описаний, выявляются и формулируются правила обработки информации и принятия решений.

Составляется список данных, требуемых для выполнения каждой из производственных или управленческих функций, а также формулируются явные и неявные правила, определяющие, как и когда выполняется каждая функция.

Результатом данного этапа являются

1) список всех создаваемых и используемых элементов данных;

**2)** перечень прикладных задач, их характеристик и используемых в них данных;

3) список принимаемых решений в управлении организацией или процессами, а также условий и правил их принятия;

4) список возможных будущих изменений в деятельности иих влияний на принятие решений.

На этапеконцептуализации собранной информации выявляются элеме**нты** предметной области, их свойства и взаимосвязи, затем синтезируется структура концептуальной модели базы данных. Наиболее известными подходами концептуализации являются анализ сущностей и представление знаний.

Так как собранная на предыдущем этапе информация является плохо структурированной, для ее концептуального анализа необходимо использовать методы системного и классификационного анализа, являющиеся универсальными инструментами организации неформализованного знания.

На первом шаге анализа ПО-информацин предметная область разбивается на несколько относительно слабо связанных между собой подобластей. Связи между элементами внутри каждой подобласти являются сильными и реализуют логические отношения типа «род - вид», «целое - часть».

Следующим шагом анализа является декомпозиция каждой подобласти, вычленение компонент (подсистем, частей) и видов (подклассов), связанных такими же отношениями с объектами соседних уровней иерархии.

Декомпозиция должна

• быть направлена на выделение элементов предметной области, существенных с точки зрения прикладных задач базы данных;

• приводить к вычленению элементов, свойства которых могут быть описаны с помощью собранных на первом этапе элементов данных;

• прекращаться при достижении уровня иерархии, исчерпывающего собранную априорную информацию об элементах данных.

Следующим шагом концептуального анализа ПО является анализ существенных свойств и взаимосвязей элементов, выделенных на стадии декомпозиции, а также формулирование и наполнение этих свойств с помощью выявленных ранее элементов данных. При этом собственные свойства элементов рассматриваются как атрибуты, а взаимосвязи между элементами - как k-арные отношения.

Синтез информационной структуры концептуальной модели проводится как композиция (сборка) структуры с учетом связей между частями.

В последние годы мощным инструментом организации плохо структурированных знаний и построения концептуальных моделей сложных предметных областей стал, так называемый, системный анализ. В основе системного анализа лежат принципы системного подхода, являющиеся методологией современного социально-научного познания и социальной практики, составляющие трактовку любого рассматриваемого объекта как системы. Системой называется совокупность элементов, находящихся в существенных отношениях и связях друг с другом. Существенность связей означает, что совместно элементы системы приобретают новое свойство (или функцию), которым не обладает ни один из элементов сислемы в отдельности. Этим система отличается от сети - совокупности элементов, несущественно связанных между собой, свойства или функции любого элемента сети не зависят от других ее элементов. Появление нового свойства системы, не сводимого к свойствам ее элементов, выражается понятием эмерджентности (целостности).

В качестве основных принципов системного подхода при построении моделей используются

• рассмотрения объекта с различных точек зрения, выявления аспектов изучаемого объекта с учетом их взаимосвязи;

• расчленения объекта на более простые подсистемы (основанием для введения подсистем является то, что связи между подсистемами много слабее, чем между элементами внутри подсистемы, а каждая подсистема много проще, чем вся система в целом);

• выделения иерархических отношений типа «целое - часть» между компонентами системы разных уровней и отношений эквивалентности между компонентами одного уровня.

Системный анализ - это применение системного подхода при обработке конкретной информации и принятию решений. Рассмотренные принципы системного подхода являются и принципами системного анализа.

Их дополняют следующие специфические принципы:

• анализ любого процесса принятия решения должен начинаться с выявления и четкой формулировки целей (желаемых результатов деятельности), которые часто определяются на основе рассмотрения системы более высокого уровня;

• необходимо рассматривать лишь те цели, вероятность достижения которых *р>р0* за время *l<t0,*где *p0* и *t0 -* пороги осуществимости цели.

Данные специальные принципы предполагают некую системную стратегию анализа, требующую рассмотрения не только самой системы, но и внешней по отношению к ней среды (надсистемы или метасистемы), и определение границы между ними.

Перспективой развития документальных информационно-поисковых систем и баз данных являются банки знаний, новая концепция информационной системы, использующая результаты исследований и разработок в области искусственного интеллекта.

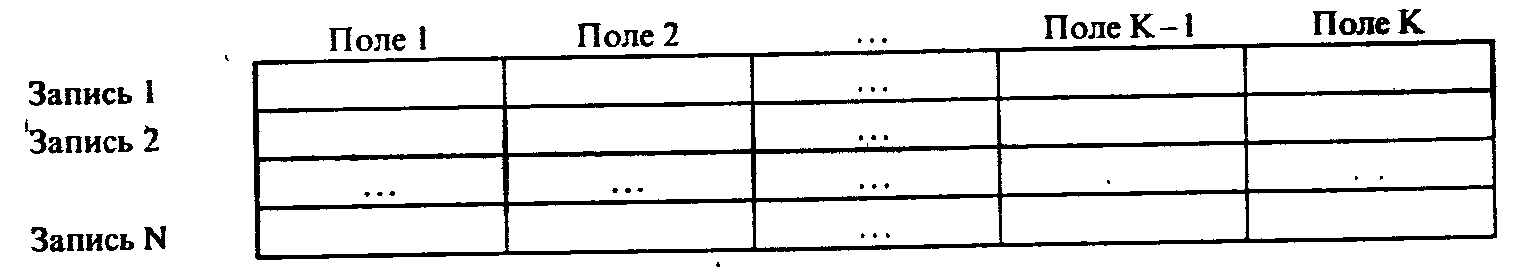
### 2.3. ПРЕДСТАВЛЕНИЕ ОБ ЯЗЫКАХ УПРАВЛЕНИЯ РЕЛЯЦИОННЫМИ БАЗАМИ ДАННЫХ ТИПА dBASE

**2.3.1. Основные элементы СУБД типа dBASE**

Напомним некоторые термины, используемые в теории реляционных баз данных. Будем представлять информацию, составляющую базу данных, в виде совокупности прямоугольных таблиц.

*Поле -* наименьший поименованный элемент информации, хранящейся в БД и рассматриваемой как единое целое (рис. 6.7).

Поле может быть представлено числом, буквами или их сочетанием (текстом). Например, в телефонном справочнике полями являются фамилия и инициалы, адрес, номер телефона, т.е. три поля, причем все текстовые (номер телефона также рассматривается как некоторый текст).



*Рис. 6.7.* Представление информации в реляционной БД

*Запись -* совокупность полей, соответствующих одному объекту. Так, абоненту телефонной сети соответствует запись, состоящая из трех полей.

Файл - совокупность связанных по какому-либо признаку записей (т.е. отношение, таблица). Таким образом, в простом случае база данных есть файл.

Все данные в БД разделены по типам. Говоря точнее, вся информация полей, принадлежащих одному столбцу (домену), имеет один и тот же тип. Такой подход позволяет ЭВМ организовать контроль вводимой информации.

При определении полей БД каждое поле может быть отнесено к одному из следующих типов:

• *символьный тип,* обозначение С (character);

• *числовой тип,* обозначение N (numeric);

• *логический тип,* обозначение L (logical);

• *тип даты,* обозначение Data;

• *примечаний,* обозначение Memo.

Каждый тип служит для хранения специфической информации. В поле типа С можно помещать любые имеющиеся на клавиатуре символы: буквы, цифры, знаки препинания и т.д. Символьное поле хранится как текст, выровненный по левому краю. Над символьными полями нельзя выполнять каких-либо вычислений.

В числовом поле могут быть только цифры, знаки «плюс» и «минус», десятичная точка. Числа выровнены по правому краю. Над числовыми полями можно выполнять вычисления.

Поле логического типа содержит величины, которые могут принимать только одно из двух значений: истинно или ложно.

Поле типа даты может содержать реальные даты в представлении, принятом в США (месяц/день/год) или в европейских странах.

Описание всех имен, типов и размеров полей составляют структуру базы данных, которая хранится в памяти вместе с данными. Эту структуру всегда можно просмотреть и при необходимости исправить. Следует различать структуру файла и информацию, содержащуюся в нем. Основное различие между структурой и информацией состоит в том, что структура остаетсянеизменной, а информацияменяетсяпри каждом редактировании или вводе.

**Семейство СУБД dBASE.** Среди СУБД для ПК наиболее распространенным в мире является семейство dBASE фирмы «Ashton-Tate». Такая популярность объясняется отчасти исторически, так как эта СУБД появилась одной из первых на рынке программных продуктов данного профиля; не в меньшей степени этому способствовала удачность заложенных основ конструкции СУБД и командного языка. Одна из наиболее широко используемых (в настоящее время устаревших) версий этого семейства - dBASE III+.

В новой версии dBASE IV концепция СУБД и командный язык получили значительное развитие, позволившее устранить недостатки предыдущей версии, а также существенно обогатить палитру средств обработки данных и улучшить интерфейс пользователя.

К числу СУБД реляционного типа, предназначенных для ПК, относятся семейство dBASE (dBASE II, III, III+ и *IV),* Clipper, Foxbase, R:BASE, Paradox и др. В этих СУБД записи и, соответственно, поля имеют обычно фиксированную длину (чаще всего длина такой записи достигает 4000...5000 байт). Исключение составляют поля типа Memo, используемые, например, в СУБД семейства dBASE. Число полей в перечисленных СУБД варьируется от 128 до 1024.

Длина поля зависит от типа поля и может составлять от 255 до 4000 байт для текстовых полей, до 20 байт для числовых полей и имеет фиксированные значения для полей типа даты (8 байт) и логических полей (1 байт). Поле типа Memo служит для хранения больших массивов текстовой информации и хранится в отдельном файле БД, но воспринимается как поле в составе основного файла БД. Это поле имеет плавающую длину, определяемую объемом введенной текстовой информации, и может достигать 32767 байт.

Большинство СУБД реляционного типа для ПК позволяют создавать файлы с числом записей до 1 млрд. и объемом до 2 Гбайт. Обычно ограничением на число и объем записей, в первую очередь, выступает емкость внешней памяти ПК на магнитных дисках.

В состав многих СУБД, предназначенных для работы на ПК, входят три основных компонента: командный язык, интерпретирующая система или компилятор для преобразования команд к выполнимому виду и средства взаимодействия пользователя с СУБД (интерфейс пользователя).

*Командный язык* служит для выполнения и обеспечения требуемых операций с данными - позволяет создавать структуры файлов БД и манипулировать данными, создавать прикладные программы, экранные формы ввода и вывода информации и т.д. Структура и возможности языка в значительной степени определяют облик конкретной СУБД, ее возможности. В состав командного языка ПК также обычно входят специальные команды по установке параметров и состояний системы (SET-команды), а также функции, предназначенные для различных видов обработки данных и выполнения вспомогательных действии.

В СУБД команды можно выполнять по одной (после набора с клавиатуры) или группами команд в автоматическом режиме, предварительно записанных в специальный (программный) файл. Команды языка СУБД записываются в текстовой форме, близкой к обычному языку. Для того чтобы ПК могла выполнять такую команду, ее надо преобразовать в вид исполнимых машинных команд.

Существует два принципиально различных способа такого преобразования. В первом способе используется *интерпретирующая система,* которая *преобразует поочередно команды* в исполнимый вид передих непосредственным выполнением. Во втором способе сначала *вся исходная программа преобразуется* (компилируется) в программу из исполнимых машиной команд и затем эта программа выполняется.

Первый способ выполнения команд СУБД имеет то преимущество, что исходная программа занимает сравнительно немного места и памяти. Кроме того, этот способ позволяет выполнять команды по одной в режиме ввода с клавиатуры или в режиме, управляемом с помощью меню. Однако выполнение исходной программы посредством интерпретации команд происходит достаточно медленно. Второй способ в отличие от первого позволяет выполнять программу значительно быстрее, но программа, составленная из машинных команд, занимает значительно больше оперативной памяти.

Взаимодействие пользователя с СУБД, в которой используется интерпретирующая система, обычно может осуществляться в режиме, управляемом с помощью меню, и реже, ввода команд с клавиатуры. Первый режим дает возможность пользователям работать с пакетом СУБД, не зная командного языка. Содержание выполняемых команд записывается в позициях меню на естественном языке. Пользователь выбирает нужную позицию меню и нажимает клавишу исполнения команды. Обычно в меню включают достаточно широкий круг команд языка СУБД, позволяющих выполнять многие операции по обработке данных, однако далеко не все. При использовании режима ввода команд с клавиатуры требуется знать их синтаксис и способы применения. Этот режим применяется уже более подготовленными пользователями. Признаком данного режима СУБД служит какой-либо знак (например, точка в левом нижнем углу экрана), который служит приглашением для набора команды. В дальнейшем для простоты будем называть этот режим просто *командным.*

Последние версии таких СУБД, как dBASE, RBASE и др., содержат генераторы прикладных программ, позволяющие пользователям без знания командного языка создавать программы средней степени сложности. Эти средства могут применяться как неподготовленными пользователями, так и профессиональными программистами для уменьшения трудозатрат при разработке прикладных программ.

Как уже отмечалось, первой созданной в этом классе программой была dBASE II, созданная Уэйном Рэтлиффом (США) в 1981г. Варианты этой системы до сих пор используются для целей обучения на учебных ЭВМ. Вскоре эта система была усовершенствована, а ее возможности расширены. Появились СУБД dBASE III и dBASE III+, ставшие на долгое время стандартом систем управления базами данных для персональных компьютеров. Позже получили распространение системы FoxBase, FoxPro и Clipper - все имеющие набор команд, похожий на СУБД семейства dBASE, и использующие ту же организацию данных в файлах с типичным расширением ***.dbf*** (Data Base File - файл базы данных), СУБД dBASE IV. Языки этих систем послужили основой для создания огромного числа прикладных информационно-поисковых систем, автоматизированных рабочих мест (АРМ) в различных предметных областях.

Система управления базами данных dBASE IV значительно усилена по сравнению с dBASE III+ по многим параметрам (ниже в скобках указаны данные для dBASE III+). В частности, можно создавать двумерные массивы переменных общим объемом до 1170 элементов, может быть открыто до 99 (16) файлов, увеличено максимальное число переменных - 15000 (265), число полей в файлах БД - 255 (128), число открытых индексных файлов -10 (7) и т.д. Как уже отмечалось выше, командный язык dBASE IV значительно расширен и улучшен по сравнению с командным языком dBASE III+.

Включение в язык dBASE IV новых команд позволило существенно улучшить следующие функции пакета СУБД семейства dBASE:

* разработку иерархических систем меню прикладныхзадач;
* разработку экранных форм ввода и вывода данных;
* организацию ввода и вывода информации в отдельные окна на экране;
* вывод данных на печать;
* возможности по поиску и обработке данных в файлах БД;
* обеспечение работы в среде ЛВС и др.

Команды типичной СУБД для персональных компьютеров предназначены для выполнения следующих функций:

• описания БД;

• наполнения БД;

• редактирования БД;

• поиска информации в БД;

• формирования и вывода на экран или принтер информации в виде отчетов.

Представление о возможностях пакета dBASE IV можно получить при знакомстве с управляющим центром (УЦ). реализующим ***интерфейс пользователя,*** управляемый меню. В состав УЦ входят шесть меню:

• меню Data («Данные») для создания, изменения структуры файлов БД, ввода, обновления и отображения данных на экране;

• меню Queries («Запросы») для формирования и выполнения запросов на выборку и обновление данных;

• меню Forms(«Формы») для разработки экранных форм ввода и вывода информации;

• меню Reports («Отчеты») для разработки форм отчетов и вывода их на печать;

• меню Labels («Марки») для разработки форм марок и вывода их на печать;

• меню Applications («Прикладные программы») - генератор разработки прикладных программ.

Функции УЦ и расположение меню на экране отражают последовательность работы пользователя при разработке прикладной информационной задачи. Работа обычно начинается с создания структуры файлов БД, которые требуются для ее решения. Возможные типы файлов приведены в табл. 6.1.

После создания структур файлов БД прикладной задачи требуется ввести в них данные.

Ввод и обновление данных в меню «Данные» выполняется на экране в стандартных формах. Эти формы не всегда вполне удобны для пользователя. В УЦ предусмотрено меню «Формы», позволяющее разрабатывать собственные экранные формы для ввода и вывода информации.

Таблица 6.1

**Типы файлов и их расширения**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Файл | Окно УЦ | Файл разработки | Файл кодов | Исполнимый файл |
| БД | Data | .dbf | - | - |
| Запрос на выборку | Queries | .qbe | - | .qbo |
| Запрос на обновление | » | .upd | - | .upo |
| Выборка dBASE III+ | » | .vue | - | - |
| Экранная форма | Forms | .sqr | .fmt | .fmo |
| Отчет | Reports | .frm | .frg | .fro |
| Марка | Labels | .lbl | .lbg | .lbo |
| Прикладная программа | Applications | - | .prq | .dbo |
| SQL-про грамма | » | - | .prs | .dbo |
| Програма, созданная генератором прикладных программ | » | .app | .prg | .dbo |

Отбор информации из одного или нескольких файлов БД можно выполнить с помощью меню «Запросы». Отобранные данные можно выводить на экран в разработанных ранее формах или на печать.

Меню «Запросы» позволяет также связывать между собой несколько файлов БД, что дает возможность при разработке структуры файлов БД прикладной задачи избежать дублирования информации.

Завершающий этап процесса обработки информации - вывод полученных данных на экран, принтер или внешние носители. Для генерации форм выходных документов предназначено меню «Отчеты».

Таким образом, меню УЦ позволяет выполнять все этапы процесса обработки информации, начиная от ее записи на носители и завершая выдачей необходимых выходных документов.

Панель УЦ состоит из шести окон. Окно Data содержит имена файлов БД. Окно Queries показывает имена файлов запросов на выборку и обновление данных. В окно Forms высвечиваются имена файлов экранных форм ввода и вывода информации. Окно Reports содержит имена файлов форм отчетов, а окно Labels - имена файлов форм марок.

Крайнее справа окно Applications высвечивает имена программ или любых текстовых файлов.

Для создания файлов соответствующих типов в окнах панели УЦ имеется позиция <Create> (создать). После выбора этой позиции высвечиваются экраны, предназначенные для выполнения операций по созданию файла соответствующего типа. Эти же экраны раскрываются и при выборе имени файла в окне.

Помимо основных функций работы с файлами различных типов управляющий центр обычно имеет широкий набор сервисных функций, позволяющих работать с каталогами, определять макропоследовательности клавиш, выполнять экспорт и импорт данных в файлы других систем, защищать данные, устанавливать параметры системы, копировать, переименовывать, удалять файлы, запускать DOS-программы и т.д.

**2.3.2. Создание структуры файлов базы данных**

Рассмотрим теперь,как создать структуру файла БД с помощью УЦ. Для этого надо выбрать позицию <Create> окна Data на панели УЦ. На экране появится таблица, состоящая из шести колонок с именами: порядковый номер поля (Num), имя поля (Field Name), тип поля (Field Type), ширина поля (Width), количество дробных десятичных знаков числа (Dec), признак индексирования поля (Index). Каждая строка этой таблицы содержит информацию об одном поле записи файла БД. Имя поля может включать до 10 буквенно-цифровых символов, но должно начинаться обязательно с буквы. Тип поля (символьный - Character, числовой с фиксированной точкой - Numeric, числовой с плавающей точкой - Float, текстовый - Memo, логический - Logical и типа даты - Date) можно выбрать клавишей ПРОБЕЛ или набрать непосредственно с клавиатуры.

После того как введено описание для всех полей файла БД, созданную структуру следует сохранить на диске. Для этого надо сначала клавишей F10 вызвать меню экрана создания/модификации структуры БД и затем выбрать позицию <Save this database file structure> (сохранить структуру файла БД) меню Layout.

Ввод новых и обновление существующих записей выполняется в одних и тех же стандартных экранных формах Browse и Edit. Предварительно надо файл БД открыть. Факт открытия файла отмечается на панели УЦ тем, что его имя перемещается выше черты под слово <Create>. Находясь на экране разработки структуры файла БД, после ее окончания можно сразу перейти к вводу данных посредством выбора позиции <Enter records from keyboard> меню Append.

Кроме того, ввод данных в стандартные экранные формы можно выполнитьизУЦ. Для этого после установки курсора на имя файла в окне Data надо нажать клавишу ENTER и затем выбрать позицию <Display data>. После этого высветится одна из стандартных экранных форм: Browse или Edit.

В форме Browse данные высвечиваются в табличном виде (строка таблицы - запись файла), при этом не все поля могут одновременно разместиться на экране.

В форме Edit поля одной записи размещаются одно под другим на одной или нескольких страницах экрана (в зависимости от количества полей). Раскрытие последующих страниц выполняется клавишей PgDn.

После того как создана структура файла БД и введена информация, обычно требуется отбирать и выводить на экран или на печать те данные, которые отвечают поставленной задаче. Для этого формируются запросы на отбор необходимых данных.

Запросы можно создавать с помощью шаблона файла (запросы по образцу) и в специальном окне, которое размещается в окне Queries панели УЦ.

Формирование шаблона вывода целесообразно выполнятьв два этапа: сначалавводить сразу все поля файла и затем удалять ненужные.

Кроме обычных полей в шаблон вывода можно помещать вычисляемые поля.

В пустую графу шаблона вычисляемых полей вводится с клавиатуры вычисляемое выражение, которое включает одно или несколько имен полей файла БД, и операторы выполнения действий с ним. В число этих операторов входят арифметические и логические операторы, а также функции dBASE.

Имеется возможность разрабатывать собственные формы ввода и вывода данных на экран. В этих формах поля файлов БД можно размещать в любом месте экрана, сопровождая их пояснительными надписями, обводить в рамки и вводить другие оформительские элементы.

Разработанные формы представляют собой программные файлы, которые можно записать на диск и затем использовать многократно для просмотра и обновления информации на экране в удобном для пользователя виде.

Разработка экранной формы начинается с выбора позиции <Create> окна Forms панели УЦ. После этого высвечивается горизонтальное меню, включающее позиции <Layout>, <Fields>, <Words>, <Goto>. Выбор этих позиций приводит к открытию различных меню, позволяющих выполнять определенный круг действий по созданию экранной формы.

Так, меню Layout предназначено для общего оформления экранной формы, реализации короткого способа ввода всех полей на экран и назначения файла БД, если он не был открыт ранее.

Добавление полей в экранную форму по одному осуществляется с помощью позиции <Add field> меню Fields.

Разработка отчетов является завершающим этапом технологии обработки информации. При выдаче отчетов могут использоваться выборки данных, полученные в результате выполнения запросов, и исходные файлы БД.

Разработка отчетов выполняется в окне Reports УЦ. Форма отчетов создается на экране. В нее можно включать следующие элементы: поля из текущего файла БД или выборки, вычисляемые поля, которые создаются специально для отчетов, текстовый материал, рамки и линии. Текст длинных символьных полей можно размещать и разворачивать в колонках заданной ширины.

В число стандартных частей отчета входят: заголовок страницы, введение (заголовок) отчета, собственно отчет (тело отчета), итоги (заключение) отчета и подножие страницы.

Описанный УЦ позволяет выполнять многие операции без знания командного языка dBASE IV с помощью режима управляемого меню, однако основной режим работы СУБД основан на использовании командного языка.

**2.3.3. Командный язык СУБД**

Операции СУБД реализованы в виде команд, которые, как правило, можно использовать независимо, вводя их с помощью клавиатуры - это, так называемый, командный режим СУБД. Из этих команд можно составлять прикладные программы, создавая таким образом системы «под ключ», ориентированные на неподготовленного пользователя. Это, так называемый, программный (пакетный) режим работы в СУБД.

В состав элементов командного языка входят команды, SET-команды, функции, переменные и массивы.

*Команда* записывается в виде строки слов, в которую входит глагол - основное слово, определяющее содержание операции, выполняемой командой, список выражений и ключевые слова с аргументами. Структура командной строки называется синтаксисом команды. Многие команды обработки файлов строятся по следующей форме:

<глагол> [<список выражений:»] [<границы>]

[FOR<условие>] [WHILE<\условие>]

[ТО FILE <имя файла> /TO PRINTER

/TО ARRAY <имя массива>

/ТО <переменные>

[ALL [LIKE/EXCEPT <шаблон>]] [IN <алиас>]

Квадратные скобки означают использование заключенных в них элементов синтаксиса по выбору. В угловых скобках, следующих после ключевого слова, записываются его аргументы. Косая черта, наклоненная влево (/), обозначает, что может быть записано только одно из разделенных этим знаком ключевых слов. Угловые, квадратные скобки и косая черта в записи команды для ее выполнения не указываются и служат только для пояснения синтаксиса.

<Список выражений> представляет собой комбинацию имен полей файлов БД, переменных, элементов массивов, констант, функций, системных переменных и операторов. Все данные в выражении должны быть одного типа. Если в выражении требуется использовать данные разных типов, их надо преобразовать в один тип специальными функциями. Выражения в списке могут быть различного типа.

Операторы в выражениях обозначают операции с данными и обозначаются специальными символами. В dBASE используются четыре типа операторов: математические, операторы отношений, логические и строковые.

Границы в приведенном выше синтаксисе команды указывают те записи файла БД, которые будут доступны при его обработке. В качестве ключевых слов границ могут быть: RECORD <n> - доступна только одна запись с заданным номером; NEXT <n> - доступны все записи, начиная с номера n; ALL - доступны все записи файла БД; REST - доступны записи, начиная с текущей до конца файла.

Опции FOR и WHILE определяют условия отбора записей для обработки. Эти опции имеют приоритет перед границами при определении доступа к записям. Опция FOR определяет доступ к обработке, начиная с первой записи и до конца файла БД, a WHILE - начиная с текущей записи и до тех пор, пока условие имеет логическое значение «Истина» (True или .Т.).

Опция ТО управляет выводом информации, обработанной командой. В ряде команд выходной поток можно направить в файл на диске, на принтер, в массив (или переменную).

Опция ALL LIKE позволяет отобрать или, наоборот, исключить среди имен файлов, полей файлов БД и переменных те, которые соответствуют заданному шаблону.

Особой разновидностью команд являются *SET-команды,* устанавливающие параметры среды, в которой выполняются обычные команды. Применяются две формы этих команд:

SET <параметр> ТО <выражение> и

SET <параметр> ON/OFF

Первая форма устанавливает значение параметра, а вторая включает (ON) или выключает (OFF) его действие, т.е. устанавливает состояние. Эти команды можно выполнять в программах, в командном режиме, можно также занести их в специальный файл с именем Config.db, определяющий параметры среды на весь сеанс работы с пакетом dBASE.

Создание файла БД осуществляется командой CREATE <имя файла>. Для модификации созданной структуры файла используется команда MODIFY STRUCTURE.

Для открытия файла БД используется команда

USE [<имя файла БД>/?] [IN <номер рабочей области>]

[INDEX <список .ndx и .mdx файлов>]

[ORDER [TAG] <имя .ndx файла>/<указатель .mdx файла>

[OF <имя .mdx файла>]]

[ALIAS<aлиac>] [EXCLUSIVE] [NOUPDATE],

которая открывает существующий файл БД и связанные с ним файлы .mdx и .ndx в текущей рабочей области.

Для создания индексных файлов типа .ndx и .mdx используется команда

INDEX ON <индексное выражение>

ТО <имя .ndx файла>/

TAG <имя указателя индекса .mdx файла>

[OF <имя .mdx файла>] [UNIQUE] [DESCENDING]

Для открытия индексных файлов используется команда

SET INDEX TO [/?<список имен файлов>

[ORDER [TAG] <нмя .ndx файла>/

<имя указателя индекса>

[OF <имя .mdx файла>]]]

Помимо широкого набора команд типичный язык СУБД типа dBASE персонального компьютера включает несколько десятков или даже сотен встроенных *функций.* Функции могут использоваться на месте переменных и констант в командах. После указания имени функции обязательно идут круглые скобки, даже если функция не требует никаких параметров. В табл. 6.2 приведены некоторые функции, часто используемые при разработке программ.

Таблица 6.2

**Некоторые функции СУБД типа dBASE**

|  |  |
| --- | --- |
| Функция | Назначение |
| RECNO()  BOF() EOF()  RECCOUNT()  FOUND()  LIКЕ(<образец>, <выр.С>)  UPPER (<выр.С>)  INKEY( )  ERROR()  CHR(<выр.N>)  DATE()  ROW() СОL() | Возвращает номер текущей записи открытого файла БД  Возвращают логический признак начала и конца открытого файла БД  Возвращает число записей в открытом файле БД  Возвращает логический признак успешности поиска в БД командой SEEK  Сравнивает строку <образец> с символьным выражением <выр.С>  Преобразует символьную строку в верхний регистр  Возвращает код нажатой клавиши  Возвращает номер ошибки  Возвращает букву по коду  Возвращает текущую дату  Возвращает номер текущей строки и столбца экрана |

В языке dBASE IV используются также функции, которые (наряду с командами) позволяют обрабатывать данные (в том числе, выполнять вычисления), а также определять состояния, например, находится ли указатель записей в конце файла: ?EOF(). После имени функции обычно стоят круглые скобки, в которых могут помещаться аргументы Функции нельзя использовать самостоятельно, а только вместе с какой-либо командой.

Кроме системных функций в dBASE IV можно использовать функции, созданные самим пользователем.

В языке dBASE используются *переменные,* тип которых автоматически определяется первым в данной программе присвоением им значения. Возможные типы: символьный, логический, числовой, дата.

Кроме простых переменных, в программах и процедурах можно использовать одно- или двумерные *массивы.* Каждый элемент массива представляет собой обычную переменную. Представление переменных в виде массива позволяет значительно расширить диапазон применения переменных, а также, в ряде случаев, облегчить реализацию функций, присущих переменным. Перед использованием массивов сначала их надо объявить командой

DECLARE <имя массива 1>

[{<количество строк>,} {<количество колонок>}]

{,<имя массива 2>

[{<количество строк>,} {<количество колонок>}]...]

В синтаксисе этой команды опции указываются фигурными скобкамивместообычно принятых квадратных. При записи команды в программе в квадратные скобки заключаются параметры (размерность) массива - число строк и столбцов. Если указано одно число, то массив одномерный. Формально двумерный массив с одним из параметров, равным единице, фактически является одномерным: DECLARE [12,1] или DECLARE [1,25].

*Примеры.*

DECLARE Mas [8,4] && Двумерный массив

DECLARE Masi [10] && Одномерный массив

STORE {15/10/89} TODatI [4,4].]

Для ввода данных в переменную или элементы массива в программе используется команда

STORE <выражение>ТО <список переменных>/<список элементов массива> или альтернативная форма

<переменная>/<элемент массива> = <выражение>.

**2.3.4. Ввод данных в базу и редактирование**

После того как создана структура файла БД, можно вводить данные. Существуют две стандартные экранные формы ввода (и вывода) информации в файлы БД: BROWSE и EDIT. Первая форма вызывается командой

BROWSE [NOINIT] [NOFOLLOW] [NOAPPEND] [NOMENU]

[NOEDIT] [NODELETE] [NOCLEAR] [COMPRESS]

[FORMAT] [LOCK <Bыp.N>] [WIDTH <выр.N>]

[FREEZE <имя поля>] [WINDOW <имя окна>]

[FIELDS <имя поля 1> [/R] [/<ширина колонки>]

/<вычисл. имя поля 1> = <выражение 1>

[ ,<имя поля 2> [/R] [/<ширина колонки>]

/<вычисл. имя поля 2> = <выражение 2>]...]

В команде имеется ряд опций, позволяющих устанавливать различные режимы ее использования. В число этих опций входит группа опций, позволяющих назначить режимы работы с полями, и группа опций по размещению полей на экране. В состав первой группы входят следующие опции: NOEDIT , позволяющая установить режим только чтения для всех полей, NOAPPEND, не разрешающая добавлять новые записи, и NODELETE, не разрешающая помечать записи для удаления. Опция FREEZE назначает для обновления только одно указанное поле. Другие поля высвечиваются, но остаются недоступными для редактирования. Если для указанного в опции FREEZE поля командой PROTECT открыто только чтение, то обновление не разрешается. Опция FREEZE без имени поля отменяет ранее назначенную установку.

Поля высвечиваются в том порядке, который определен структурой файла БД или указан в команде опцией FIELDS, входящей в состав второй группы. Эта опция позволяет выбрать поле и определить порядок его расположения в таблице Browse.

Опция WIDTH устанавливает верхний предел ширины колонки для всех полей в таблице BROWSE. Эта ширина перекрывает ширину поля, назначенную при создании файла БД. Если опции WIDTH и <ширина колонки> используются одновременно, то принимается наименьшее значение. Данная опция, как и предыдущая, не применяется для полей типа Memo и логических полей. Значения числовых полей и полей даты не будут высвечиваться, если значение опции WIDTH меньше ширины этих полей, назначенных при определении структуры файла БД. Опция FORMAT позволяет разместить поля в таблице BROWSE в очередности, определенной экранной формой, которая установлена командой SET FORMAT TO.

Альтернативой команды BROWSE является команда EDIT

EDIT [NOINIT] [NOFOLLOW] [NOAPPEND] [NOMENU]

[NOEDIT] [NODELETE] [NOCLEAR] [<номерзаписи>]

[FIELDS <спнсок полей>] [<диапазон>]

[FOR <условие>] [WHILE <условие>]

Эта команда выводит на экран по одной записи файла БД, поля которой размещаются по вертикали.

Если требуется обновить содержимое полей файла БД без привлечения данных из другого файла, например умножить содержимое числового поля на заданную 'величину, соединить содержимое двух символьных полей и т.д., можно использовать команду

REPLACE [<границы>] <поле> WITH <выражение>

[, <поле> WITH <выражение...]

[WHILE <условие>] [FOR <условие>] [ADDITIVE]

которая изменяет (замещает) значения заданных полей (включая поля Memo) текущей записи или группы записей, определенных границами и опциями WHILE, FOR.

После ввода данных в файл БД нередко требуется быстро их просмотреть. Для этого служат команды

LIST/DISPLAY [<границы>] [[FIELDS] <список выражений>]

[OFF] [<диапазон>] [FOR <условие>]

[WHILE <условие>] [ТО PRINTER/

ТО FILE <имя файла>]

Эти команды выводят все поля файла БД, если их список не определен опцией FIELDS. В начале каждой записи, если не указана опция OFF, выводится ее номер, опции <диапазон>, FOR и WHILE определяют диапазон выводимых записей. Команды LIST и DISPLAY идентичны с той разницей, что первая выводит записи непрерывно, а вторая делает остановку после вывода одной страницы экрана.

Кроме выводов файлов БД команды LIST и DISPLAY позволяют также высвечивать данные о файлах в директориях, содержимое буфера HISTORY (хранилище выполненных с клавиатуры команд), данные об используемых переменных и массивах, информацию о текущем сеансе работы с dBASE, структуру указанного файла БД и имена рабочих станций, работающих в среде ЛВС.

**2.3.5. Дополнительные операции**

Нередко возникает необходимость *просмотреть структуру файла* и вывести ее на печать. Для этого служит команда

LIST/DISPLAY STRUCTURE [IN <алиас>] [ТО PRINTER/TO FILE <имя файла>]

которая выводит информацию о файле БД.

Существуют специальные команды перемещения по файлу, добавления новых записей и пометки записей для их последующего удаления.

Для *перемещения по файлу* используется команда

GO/GOTO BOTTOM/TOP [IN <алиас>]

или

GO/GOTO [RECORD] <номер записи> [IN <алиас>]

или

<номер записи>

Эти команды осуществляют переход на указанную запись в активном файле БД.

В программах часто требуется *перемещать указатель записи* не на конкретную запись, а на следующую после текущей. Эту операцию выполняет команда

SKIP [<Bыp.N>] [IN <алиас>]

После того как указатель записи перемещен на нужную запись, которая становится текущей, можно выполнить требуемую операцию, например ввести запись до или после текущей. Для этого используется команда

INSERT [BEFORE] [BLANK]

которая позволяет ввести новую запись (или пустую запись, если указана опция BLANK) после текущей или перед ней (если задана опция BEFORE). Пометить одну или несколько записей для их последующего удаления можно командой

DELETE [<диапазон>] [FOR <условие>] [WHILE <условие>]

Опции <диапазон>, FOR и WHILE позволяют *помечать записи,* отобранные в соответствии с заданным диапазоном и условиями.

Для того чтобы *уничтожить пометки* на удаление записей используется команда

RECALL [<диапазон>] [FOR <условие>] [WHILE <условие>]

Физическое удаление помеченных записей активного файла БД выполняет команда PACK. Команда

APPEND [BLANK]

позволяет *добавлять записи,* введенные с клавиатуры в конец активного файла БД.

В ряде случаев требуется *создать физически упорядоченный файл.* Для этого используется команда

SORT TO <имя файла> ON <поле 1 > [/А] [/С] [/D]

[ ,<поле 2> [/А] [/С] [/D]...] [ASCENDING]/[DESCENDING]

[<границы>] [WHILE <условие>] [FOR <условие>]

которая создает новый файл БД с записями, отсортированными в алфавитном, хронологическом или числовом порядке по заданным полям. Сортировка выполняется в порядке возрастания (/А) или убывания *(ID)* значений полей. Допускается сортировка максимум по десяти полям. Опция /С определяет сортировку независимо от прописных и строчных букв. Команда

• COPY FILE <имя файла 1> ТО <имя файла 2>

делает *копию файла* любого типа.

После копирования или в других случаях нередко требуется *переименовать файл.*

Для этого используется команда

RENAME <имя файла> ТО <новое имя файла>

Нередко требуется найти запись не по ее положению в файле БД, а по содержанию полей. Такой вид поиска предусмотрен, например, в меню команды BROWSE. Способ поиска по содержанию полей зависит от того, созданы ли индексные файлы по этим полям или по их комбинациям. Для *поиска по неиндексированным полям* используется команда

LOCATE [FOR] <условие> [<диапазон>] [WHILE <условие>]

Эта команда выполняет в активном файле БД поиск первой записи, удовлетворяющей условию, заданному опцией FOR. Опции <диапазон> и WHILE определяют диапазон и условие начала и конца просмотра записей.

Для *поиска по индексированным полям* используются команды

SEEK и FIND.

Команда

SEEK <выражение>

выполняет поиск первой записи файла БД с заданным значением индексированного поля. При этом соответствующий индекс должен быть установлен в качестве главного (управляющего) индекса. Команда

FIND <литеральная строка>

аналогична команде SEEK и также выполняет *поиск первой записи в индексированном файле БД с заданным значением индекса* при установленном соответствующем главном индексе. Отличие состоит в том, что в качестве поискового значения в команде FIND используется не выражение, а литеральная строка.

Основной командой, предназначенной для *создания нестандартных экранных форм ввода и вывода* информации, является

@<строка>,<колонка>

[SAY <выр.>

[PICTURE <выр.С>]

[FUNCTION <списокфункций>]]

[GET <переменная>

[[OPEN] WINDOW <имя окна>] [PICTURE <выр.С>]

[FUNCTION <список функций>]

[RANGE [<нижний>] [<верхний>]]

[VALID <условие> [ERROR <выр.С>]]

[WHEN <условие>] [DEFAULT <выражение>]

[MESSAGE <выр.С>]]

[COLOR [<стандартная область экрана>] [<нестандартная область

экрана>]]

Эта команда позволяет выводить или вводить информацию в заданном формате в указанные места экрана. Кроме того, с помощью опций SAY этой команды можно выводить данные на принтер или в файл на диске. Для того чтобы установить вывод информации с помощью данной команды на экран, принтер или в текстовый файл DOS, используется команда

SET DEVICE TO SCREEN/printer/file <имя файла>

Опция GET выводит и позволяет *обновлять и редактировать данные,* содержащиеся в полях, переменных или массивах.

Собственно *обновление данных* в переменных GET (здесь и нижепод словом «переменная» подразумеваются также поля файла БД и элементы массива) выполняется командой READ. которую следует поместить после данной команды. Эта команда активизирует переменные и выводит в них значения, находящиеся на экране.

Для *приостановления процесса последовательного выполнения программы* используется команда

WAIT [<сообщение>] [ТО <переменная>]

которая вызывает паузу до нажатия любой клавиши и высвечивает в нижней строке экрана сообщение. После нажатия клавиши выполнение программы продолжается, а в переменную заносится код ASCII этой клавиши.

С целью *улучшения восприятия информации*на экране можно использовать команду

@ <строка 1>,<колонка 1> ТО <строка 2>,<колонка 2> [DOUBLE/PANEL/<cимволы определения рамки>]

[COLOR <код цвета>]

которая рисует на экране прямоугольную рамку, определяемую координатами верхнего левого - <строка 1>,<колонка 1> - и правого нижнего - <строка 2>, <колонка 2> - углов рамки Опция DOUBLE позволяет рисовать двойную линию рамки (по умолчанию рисуется одинарная линия), опция PANEL - рамку в виде сплошной полосы. Кроме того, имеется возможность рисовать линию рамки заданными символами. Для этого задается список символов для определения рамки. Можно задать различные символы для сторон и углов рамки. В этом случае символы разделяются запятыми и располагаются в следующем порядке: f, b, 1, r, fl, fr, bl, br, где f - верх; b - низ; 1 - левая сторона; г - правая сторона; fl -левый верхний угол; fr - правый верхний угол; bl - левый нижний угол; br -правый нижний угол.

*Пример.*

@ 5,13 ТО 10,40 DOUBLE COLOR R/BG Д

Для *полной очистки всего экрана* используется команда

CLEAR [ALL/FIELDS/GETS/MEMORY/MENUS/

POPUPS/TYPEAHEAD/WINDOWS]

Она очищает экран, устанавливает курсор в левый нижний угол экрана и освобождает все незаконченные опции GET команды @... SAY. ..GET. Различные опции этой команды также закрывают файлы БД, очищают переменные, списки полей, окна, горизонтальные и вертикальные меню и буфер клавиатуры.

Команда @... SAY. ..GET является универсальной для *организации ввода и вывода информации* на экране. В программах нередко требуется вводить с клавиатуры данные для обеспечения процесса обработки информации, например имя файла БД, данные для ветвления программы, поисковые значения и т.д. Для этого предусмотрена специальная команда

ACCEPT [<сообщение>] ТО <переменная>

которая позволяет вводить с клавиатуры данные символьного типа в переменную. Перед полем переменной можно высветить сообщение, например, о характере вводимых данных.

Кроме этой команды для *ввода данных с клавиатуры* можно использовать команду

INPUT [<сообщение>] ТО <переменная>

которая выполняет аналогичные функции, но позволяет вводить в переменную с клавиатуры данные не только символьного, но и других типов, а также в виде заданного выражения, что не допускается в команде ACCEPT. Тип выражения определяет тип переменной, в которую вводится значение. После набора выражения с клавиатуры для ввода его в переменную надо нажать клавишу Enter.

Кроме универсальной команды SAY существует команда, специально предназначенная для *вывода данных на экран и принтер:*

?/?? [<выр.1> [PICTURE <выр.С>]

[FUNCTION <список функций>] [AT <выр.N>]

[STYLE <номер шрифта>]] [,<выр.2>...] [,]

которая выводит значение одного или более заданных символьных выражений. Если установлено состояние SET PRINT ON. то выполняется вывод на принтер, если SET PRINT OFF, то на экран. Эта команда в форме ? выводит каждое выражение на следующей строке, а в форме ?? все выражения на одной строке.

Для *печати отчета* используется команда

REPORT FORM <имя файла отчета>/?

[<границы>] [WHILE <условие>]

[FOR <условие>] [PLAIN]

[HEADING <выр С>] [NOEJECT]

[ТО PRINTER/TO FILE <имя файла>] [SUMMARY]

которая запускает программу печати отчета (файл отчета), созданную командой CREATE/MODIFY REPORT.

Для *создания исходных программных файлов* используются обычные текстовые редакторы, например встроенный редактор dBASE, который вызывается командой

MODIFY COMMAND/FILE <имя файла> [WINDOW <имя окна>]

Прикладная программа обычно состоит из главной процедуры и вызываемых из нее процедур, например:

<команды> && Начало главной процедуры

DO Proс1

DO Proc2

DO Ргос3

RETURN && Конец главной процедуры

PROCEDURE Proс1 && Начало Proс1

<команды>

RETURN && Конец Proс1

PROCEDURE Proс2 && Начало Proc2

<команды>

RETURN && Конец Proc2

PROCEDURE Ргос3 && Начало Ргос3

<команды>

RETURN && Конец Рrос3

*Процедура* представляет собой программный модуль, который выполняет определенную задачу. После ее выполнения управление передается в вызвавшую или главную процедуру. Процедура так же, как и программа, вызывается и выполняется командой

DO <имя программы>/<имя процедуры> [WITH <список параметров>]

Опция WITH передает в вызываемую процедуру параметры (до 64).

Каждая процедура, размещаемая в файле процедур или в программном файле, должна начинаться с команды

PROCEDURE <имя процедуры>

и завершаться командой

RETURN [<выражение>/ТО MASTER/TO <имя процедуры>]

Команда RETURN удаляет все переменные, которые были определены в процедуре как PRIVATE (переменные, определенные как PUBLIC, остаются без изменения).

Командой DO в процедуру можно передавать из вызывающей программы и обратно параметры, которые в самой процедуре определяются командой

PARAMETERS <список параметров>

которая должна быть первой исполняемой командой в теле процедуры. В качестве параметра в списке может быть любое допустимое выражение, определяющее переменную, в которую передаются данные из соответствующего параметра вызывающей программы. Если эта переменная не объявлена глобальной (PUBLIC), то автоматически она имеет статус локальной (PRIVATE).

Определение переменных как локальных выполняется командой

PRIVATE <список переменных> / [ALL [LIKE/EXCEPT <шаблон>]]

Глобальные переменные можно использовать повсюду а программе и в вызываемых ею процедурах. Для объявления переменных и элементов массива глобальными используется команда

PUBLIC <список переменных>/ [ARRAY <список элементов массива>]

**2.3.6. Организация системы меню**

Чтобы организовать в dBASE систему меню, сначала надо определить **его** отдельные компоненты - меню вертикального и горизонтального типа. Для этого используются специальные команды, начинающиеся словом DEFINE - определить. Определение горизонтального меню выполняется командой

DEFINE MENU <имя меню> [MESSAGE <выр.С>]

Эта команда еще не создает горизонтальное меню, а только назначает ему имя и, если требуется, сообщение, которое будет высвечиваться в нижней строке экрана (строке сообщений). Для определения позиций горизонтального меню используется команда

DEFINE PAD <имя позиции>

OF <имя меню> PROMPT <выр.С>

[AT <строка>,<колонка>] [MESSAGE <выр.С>]

которая определяет одну позицию горизонтального меню. Для задания нескольких позиций данная команда повторяется соответствующее число раз. Вертикальное меню определяется командой

DEFINE POPUP <имя меню>

FROM <строка 1>,<колонка 1>

[ТО <строка 2>,<колонка 2>]

[PROMPT FIELD <имя поля>]

/PROMPT FILES [LIKE <шаблон>]

/PROMPT STRUCTURE]

[MESSAGE <выр.С>]

которая устанавливает его имя (записывается по тем же правилам, что и для горизонтального меню) и координаты окна меню на экране. Опции FROM и ТО определяют координаты левого верхнего и правого нижнего углов окна меню.

Назначение текста и других атрибутов одной позиции ранее определенного вертикального меню осуществляется командой

DEFINE BAR <номер строки> OF <имя меню>

PROMPT <выр.С>

[MESSAGE <выр.С>]

[SKIP [FOR <условие>]]

Чтобы определить нескольких позиций, надо эту команду повторить соответствующее число раз.

После того как определены позиции меню, обычно требуется назначить им определенные действия, которые будут выполняться при их выборе. Эти действия для позиций горизонтального меню назначаются командой

ON SELECTION PAD <имя позиции горизонтального меню>

OF <имя горизонтального меню> [<команда>]

которая выполняет заданную команду при выборе указанной позиции горизонтального меню. В качестве команды может быть любая команда dBASE IV, в том числе команда вызова программ и процедур.

Для назначения действий позициям вертикального меню используется команда

ON SELECTION POPUP <имя позиции вертикального меню>/ALL [<команда>]

которая выполняет заданную команду или процедуру при выборе позиции вертикального меню.

Обычно для назначения позициям меню каких-либо действий используется выход на процедуру, в которой помещаются команды для выполнения этих действий. В такой процедуре чаще всего применяется команда

DO CASE

CASE <условие>

<команды>

[CASE <условие>]

<команды>

[OTHERWISE]

<команды>

ENDCASE

Можно формировать вложенные структуры этой команды, т.е. внутри команды после ключевого слова CASE помещать другую команду DO CASE, а также любые другие команды ветвления и выполнения циклов, например IF...ENDIF, DO WHILE...ENDDO и др.

*Пример.*

DO CASE

CASEVal> 1000

<команды>

CASEVal> 100

<команды>

CASEVal>10

<команды>

OTHERWISE

<команды>

ENDCASE

Если требуется разветвить процесс только на два направления, можно использовать команду

IF <условие> <команды> [ELSE] <команды> ENDIF

При реализации операций, назначаемых позициям меню, нередко требуется организовать циклы повторяющихся действий, например организовать обработку файла БД и др. Это можно сделать с помощью ряда команд и их комбинаций. Одной из таких является команда

DO WHILE <условие>

<команды>

[EXIT]

<команды>

[LOOP]

<команды>

ENDDO

которая обеспечивает повторение последовательности команд, заключенной между словами DO WHILE и ENDDO до тех пор, пока заданное условие истинно. Если встречается слово LOOP, то управление передается в начало цикла команд, а при встрече слова EXIT осуществляется выход из команды и управление передается на первую команду после слова ENDDO, завершающего данную команду.

Предыдущие команды позволяют определить меню и атрибуты их позиций. Для того чтобы активизировать горизонтальное и вертикальное меню (а также списки), используются команды

ACTIVATE MENU <имя меню> [PAD <имя позиции>]

и

ACTIVATE POPUP <имя меню>

которые вызывают на экран предварительно определенные меню.

**2.3.7. Пример создания информационной системы с помощью СУБД типа dBASE**

Рассмотрим пример создания информационной системы обработки школьной информации «Всеобуч» с помощью СУБД типа dBASE. Проследим основные этапы работы.

1. Постановочная часть.

1. 1. Цель функционирования ИС.

1.2. Структурирование общей задачи.

1.3. Анализ входной информации.

1.4. Анализ переработки информации.

1.5. Анализ выходной информации (структура БД, формат экрана для выходных данных, выходные формы информации, стандартные запросы, командные файлы и т.д.)

2. Создание БД.

2.1. Проектирование структуры БД.

2.2. Создание файла БД.

2.3. Заполнение файла БД записями.

3. Создание комплекса вспомогательных сервисных файлов.

3.1. Файл формата экрана.

3.2. Файлы выходных форм документов.

3.3. Индексированные файлы.

3.4. Файлы стандартных запросов.

3.5. Комплекс командных файлов.

4. Эксплуатация созданной ИС.

4.1. Поддержка информации в БД.

4.2. Получение необходимой информации через файлы запросов.

Согласно этим этапам разработаем прикладную программу, которая иллюстрирует возможности СУБД в автоматизации управления школой.

**Постановка задачи.** Цель разработки этого программного средства **-** автоматизация работы завуча средней школы по одной из форм отчета всеобуча.

Предполагается получение информации в виде табл. 6.3 и табл.6.4.

Таблица 6.3

**Информация для отчета завуча школы**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Класс | Сохранение контингента | | Неуспевающие | Второгодники, их успеваем. | Пропуски | | Работа с трудными учащимися и их семьями | |
| Конец года | Четверти 1 2 3 4 | Четверти  1 2 3 4 | Четверти 1 2 3 4 | Без ув. причин | По болезни | Состоящ.  на учете | Трудные семьи |
|  |  |  |  |  |  |  |  |  |

Таблица 6.4

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Класс | Многодетные семьи и опекаемые дети | | Горячее питание | | | Медицинское обслуживание | | Учебники, дидактические материалы | ГПД | УПК |
| семьи | опекаемые дети | фонд всеобуча | б/п молоко | 1 2  п. п. | количество осмотров | хронические заболев. |
|  |  |  |  |  |  |  |  |  |  |  |

Входной информацией в БД выступают следующие данные по каждому классу: число учеников на конец прошлого года и по четвертям; число неуспевающих в классе по четвертям; число пропусков в классе на текущий момент (по болезни и без уважительной причины. Учитывается работа с трудными семьями, медицинское обслуживание. Собирается информация о сохранности учебников и наглядно-дидактических материалов; о группах продленного дня и посещении УПК.

Требования к переработке информации:

• ввод данных и автоматическая сортировка по классам;

• добавление записей;

• внесение изменений в записи по запросу на номер класса;

• удаление записей посредством пометки знаком «\*» в режиме просмотра данных. Выходная информация:

• просмотр данных в виде табл. 6.3 или табл. 6.4, а также просмотр данных по одному из столбцов таблиц, например, только пропуски уроков;

• вывод на принтер табл. 6.3 или табл. 6.4, а также печать только определенных столбцов.

Ввод данных удобно осуществлять с использованием экранных форм, повторяющих на экране таблицы. Так как эти таблицы не помещаются на экране одновременно и содержат неповторяющуюся информацию, целесообразно разделить ввод данных на две части. Обработку информации также удобнее проводить по двум таблицам по отдельности. Выбор осуществим по меню:

1. Часть 1. 2. Часть 2. 3. Конец работы.

Создадим две БД и проиндексируем их с тем. чтобы при вводе данных они автоматически сортировались по классам и чтобы в дальнейшем ускорить поиск требуемой информации. Над каждой из двух таблиц организуем проведение следующих операций:

1) заполнение новойбазы**;**

2) добавление записей;

3)корректировку;

4)удаление записей;

5) печать документов;

6)выход.

Рассмотрим характеристики программ по каждому из представленных пунктов меню.

1) Этот пункт предполагает, что все записи БД уничтожаются и предварительно появляется предупреждение:

«При создании новой БД все данные уничтожаются. Вы действительно хотите создать новую БД? (д/н)».

При нажатии клавиши «д» пользователю предоставляетсяпустая экранная форма, соответствующая выбранной таблице из главного меню.

2) При добавлении записей на экране высветим пустую экранную форму. Пользователь может вносить информацию, передвигаясь по столбцам. При этом курсор автоматически по заполнении столбца переходит в следующий. После заполнения каждой строки в нижней части экрана высвечивается сообщение:

«Продолжить добавление записей? (д/н) :: »

При ответе «н» происходит возврат в меню, а записи сортируются по классам.

3) Для корректировки также высветим экранную форму, но с уже заполненными столбцами. Здесь необходимо предусмотреть тот случай, когда изменения должны вноситься только в отдельные классы. В этой связи требуется создание подменю.

4) При удалении записей, необходимо, как и в режиме корректировки, предусмотреть подменю с запросами по классам. Пользователю необходимо поставить знак «звездочка» (\*) в столбце «классы», о чем в нижней части экрана организуем подсказку:

«Для удаления записи, на которой стоит курсор, пометьте ее звездочкой (\* )»

По завершении работы все помеченные записи уничтожаются, а оставшиеся автоматически сортируются по классам.

5) Этот раздел служит для получения выходной информации. При его выборе создадим еще одно меню:

(Часть 1)

1) Полный отчет 2) Сохранение контингента 3) Неуспевающие

4) Успеваемость второгодников 5) Пропуски 6) Трудные учащиеся

(Часть 2)

1) Полный отчет 2) Многодетные семьи и опекаемые дети

3) Горячее питание 4) Медицинское обслуживание

5) Сохранность учебников 6) ГПД 7) УПК

Здесь по требованию пользователя должна выводиться информация либо о всех классах сразу, либо только о тех, которые укажет пользователь в ответе на запрос В этом же запросе сразу определим, на какое устройство осуществляется вывод. Вид запроса:

«Какие классы Печать (д/н)»

Полный отчет (первый пункт меню) выводится в соответствии с экранными формами табл. 6.3 и табл. 6.4. Для всех остальных пунктов создадим форматные файлы, которые выводят информацию по одному из столбцов экранных форм.

Таблица 6.5

**Отчет по «Медицинскому обслуживанию»**

|  |  |  |  |
| --- | --- | --- | --- |
| Медицинское обслуживание | | | |
| Класс | Количество осмотров | Хронические заболевания | Специальные медицинские группы |
| 1а | 30 | 3 | 1 |
| 1б | 26 | 2 | 0 |
| 1в | 30 | 0 | 0 |
| 2а | 29 | 1 | 1 |
| … | … | … | … |
| \*\* итог \*\* | 820 | 21 | 14 |

**Структура БД.** Создадим две БД со структурами, изображенными в табл. 6.6. Поле «класс» содержит информацию о номере класса. По полю «класс1» различаются параллельные классы.

Для быстрого поиска информации в базе, а также для автоматической сортировки записей после внесения изменений, проиндексируем их по полю «класс» и «класс1». При этом создаются индексные файлы KL.NDX и KL1.NDX.

Таблица 6.6

**Структуры создаваемых баз данных**

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Имя | Тип | Длина |
| 001 | класс | N | 002 |
| 002 | Класс1 | С | 002 |
| 003 | пр:год | N | 005 |
| 004 | четв:1 | N | 002 |
| 005 | четв:2 | N | 002 |
| 006 | четв:3 | N | 002 |
| 007 | четв:4 | N | 002 |
| 008 | неусп:ч1 | N | 002 |
| 009 | неусп:ч2 | N | 002 |
| 010 | неусп:ч3 | N | 002 |
| 011 | неусп:ч4 | N | 002 |
| 012 | втгод:ч1 | N | 002 |
| 013 | втгод:ч2 | N | 002 |
| 014 | втгод:ч3 | N | 002 |
| 015 | втгод:ч4 | N | 002 |
| 016 | проп.без у/п | N | 004 |
| 017 | по:болезни | N | 004 |
| 018 | на:учете | N | 005 |
| 019 | тр.семья | N | 005 |
|  |  |  |  |
| Поле | Имя | Тип | Длина |
| 001 | класс | N | 002 |
| 002 | класс1 | С | 002 |
| 003 | семьи | N | 005 |
| 004 | дети | N | 005 |
| 005 | фонд | N | 005 |
| 006 | молоко | N | 004 |
| 007 | пол1 | N | 003 |
| 008 | пол2 | N | 003 |
| 009 | осмотр | N | 005 |
| 010 | хрон | N | 005 |
| 011 | спрг | N | 004 |
| 012 | учебн | С | 010 |
| 013 | гпд | С | 003 |
| 014 | упк | С | 005 |

Создадим форматные файлы отчетов для вывода информации по одномуизстолбцов таблиц: PROP FRM (пропуски), TRUD.FRM (трудные учащиеся), UPK.FRM (УПК), SOHR.FRM (сохранение контингента), WTORG.FRM (успеваемость второгодников). NEUSP.FRM (неуспевающие), SEM.FRM (многодетные семьи и опекаемые), PIT.FRM (горячее питание), MED.FRM (медицинское обслуживание), UBN.FRM (сохранность учебников), GRD.FRM (группы продленного дня).

Ниже представлены тексты некоторых программ, поддерживающие рассматриваемую информационную систему (автор -Т.П. Глазко).

*Программа 140*

\_\_\_\_\_\_\_\_\_\_\_\_ модуль MAIN. PRG \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

erase

set colon on

set talk off

store ' ' to rl,dd,d3

do whiled3<>'3'

erase

do while dd<>'l' .and. dd <>'2' .and. dd<>'3'

@ 5,12 say ' 1). Часть 1 '

@ 6,12 say ' 2). Часть 2

@ 7,12 say ' 3). Конец работы '

@ 8,12 say ' Введите цифру ' get dd

read

loop

enddo

If dd<>'3'

do case

case dd= '1'

use newws Index kl

case dd*= '2'*

use newwsl Index kll

endcase

do menu

else

store '3' to d3

endif store ' ' to dd

loop

enddo

cancel

\_\_\_\_\_\_\_ МОДУЛЬ MENU.PRG \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MENU.PRG поддерживает меню операций и в зависимости от выбранного пункта передает управление модулям: SERVIS2.PRG (корректировка), APPN PRG (добавление), NEWBD.PRG (создание новой БД), SERVIS3.PRG (удаление), DOC.PRG (документы). Прежде чем будет вызвана программа SERVIS2.PRG, выполняется еще одна программа - DOP.PRG, которая к этому меню добавляет запрос на классы.

*Программа 141*

set format to screen

store 0 to dl, d2

do while d2<>8

goto top

erase

store 0 to dl

text

МЕНЮ

1) Заполнение новой базы 4) Удаление записей

2) Добавление записей 5) Документы

3) Корректировка 6) Выход

endtext

do while d1<>l .and. dl<>2 .and. dl<>3 .and. dl<>4; .and. dl<>5 .and.

dl<>6

@ 9,1 say ' Введите цифру' get dl picture '9'

read

loop

enddo

do case

case dl=3

do DOP

do SERVIS2

case dl=2

do APPN

case dl=l

do NEWBD

case dl=4

do SERVIS3

case dl=5

do DOC

case dl=6

store 8 to d2

endcase

loop

enddo

return

\_\_\_\_\_\_\_\_\_\_\_\_\_ модуль DOP.PRG \_\_\_\_\_\_\_\_\_\_\_\_\_\_

DOP.PRG осуществляет запрос на классы, по результатам ввода формирует логическую переменную kkk. используемую в некоторых других модулях для контроля за номерами классов, kkk принимает значение «истина», если поле «класс» соответствует введенным номерам, и «ложь» - в противоположном случае.

*Программа 142*

store 0 to kl

store to kl

store Т to pp

\*\*\* запрос на классы

@ 10,1 say 'Какие классы\*

do while pp

@ 10,13 get kl picture '99-99'

read

if val($(kl,l,2))<12 .and. val($(kl,4,2))<l2

store F to pp

endif

loop

enddo

\*\*\* Формирование логической переменной kkk

if val ($(kl,4,2))=0

store 12 to kl

else

store val($(kl,4,2))+l to kl

endif

store 'Kлacc>val ($ (kl, 1,2))—1 .and. Knacc<kl' to kkk

return

\_\_\_\_\_\_\_\_\_\_\_\_ модуль SERV1S2.PRG \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Этот модуль позволяет внести изменения в записи БД. Классы, в которых будет производиться корректировка, пользователь указывает, отвечая на запрос до вызова настоящей программы.

*Программа 143*

erase

set colon off

set talk off

goto top

if val(dd)=l

do shapka

else do shapkal

endif

store 1 to numl ,num2

store Т to log

store ' ' to rl

store 0 to pointer, num

store 't' to dl

count to num

do while num1<num .and. num2<num .and. log

store 10 to line

store 't' to dl

@ 21,25 say 'Нажмите«к>> для перехода к другим записям

@ 22,25 say "или ВВОД для внесения изменений'

if val(dd)=l

do newws2

else

do newws21

endif

skip - line + 10

store 10 to line

do i zm2

store pointer+1 to pointer

loop

if num1<num

skip

endif

enddo

\_\_\_\_\_\_\_\_\_\_\_\_\_ модуль SHAPKA.PRG \_\_\_\_\_\_\_\_\_\_\_\_\_

*Программа 144*

erase

text

таблица 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Класс | Сохранение контингента | | Неуспевающие | Второгодники, их успеваем. | Пропуски | | Работа с трудными учащимися и их семьями | |
| Конец года | Четверти 1 2 3 4 | Четверти  1 2 3 4 | Четверти 1 2 3 4 | Без ув. причин | По болезни | Состоящ.  на учете | Трудные семьи |
|  |  |  |  |  |  |  |  |  |

endtext

return .

\_\_\_\_\_\_\_\_\_\_\_\_ модуль SHAPKA1.PRG \_\_\_\_\_\_\_\_\_\_\_\_

*Программа 145*

erase

text

таблица 2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Класс | Многодетные семьи и опекаемые дети | | Горячее питание | | | | Медицинское обслуживание | | Учебники, дидактические материалы | ГПД | УПК |
|  | семьи | опекаемые дети | фонд всеобуча | б/п молоко | 1  п. | 2  п. | Количество осмотров | Хронические заболевания |