МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

САМАРКАНДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени Алишера Навои

Факультет информатики и информационных технологии

ВЫПУСКНАЯ РАБОТА

СОЗДАНИЕ БАЗЫ ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ТАЛАБА» В ИНТЕГРИРОВАННОЙ СРЕДЕ «DELPHI - 6»

,

«DELPHI - 6» ИНТЕГРАЛАШГАН МУХИТИДА «ТАЛАБА» ИНФОРМАЦИОН ТИЗИМИНИНГ МАЪЛУМОТЛАР БАЗАСИНИ ЯРАТИШ

Специальность “Прикладная математика и информатика ”

Студент 4 курса
Туракулов М.

Научный руководитель
проф.Жуманов И.И.

С А М А Р К А Н Д - 2 0 10

С О Д Е Р Ж А Н И Е

ВВЕДЕНИЕ

Непрерывное увеличение объемов информации, массовое применение персональных компьютеров, появление автоматизированных рабочих мест (АРМ) для специалистов требуют создания новых технических средств, методов и систем обработки информации.

Автоматизированные информационные системы (ИС) играют важную роль в современной информационной технологии.

Они представляют интерес с одной стороны, как самостоятельные объекты и. с другой стороны, являются основными компонентами других систем; АСУ, систем автоматизации проектирования (САПР), систем автоматизации учрежденческих работ и др.

Языково-программным ядром информационных систем является система управления базами данных (СУБД). Разработка ИС представляет собой сложную практическую задачу и имеет итеративный характер.

Целью настоящей работы является исследование структуры создания информационной системы в управление деятельностью студентов ВУЗа, технологии построения баз данных, разработка алгоритмов и программных комплексов функциональных задач студенческого отдела, а также изложение результатов реализации информационной технологии. В работе поставлены и решены следующие задачи:

1. ОПИСАНИЕ СТРУКТУРЫ ИНФОРМАЦИОННЫХ СИСТЕМ

Общепринято, что в информационных системах основной компонентой является хранимая информация. Эта информация является отображением некоторой реальной предметной области. Другими компонентами системы являются средства обработки информации, а также носитель информации. Другими словами, *информационная система* - это комплекс, состоящий из информационного фонда (базы данных), программного обеспечения (системные и прикладные программы управления, ввода, обработки и вывода информации) и носителя информационного фонда (МД, МЛ).

Классификация и способы программной реализации **систем.**

Согласно работе [1] информационные системы классифицируют в двух аспектах по их функциональным возможностям, т.е. что они могут делать и по структуре информационного фонда, то какую логическую структуру имеет база данных системы. Выделяют следующие функциональные возможности: автоматизированные и неавтоматизированные системы, системы с корректировкой и без корректировки ответа; с библиотекой и без библиотеки обрабатывающих программ и т.п. Типы логических структур информационного фонда будут рассмотрены при анализе различных подходов к организации баз данных.

Существует несколько способов программной реализации ИС, основными среди которых являются:

• автономная система, не использующая стандартное математическое обеспечение ЭВМ. При таком подходе ЭВМ используется только для информационных задач, т.е. только для реализации ИС. Математическое обеспечение конкретной ЭВМ в этом случае состоит из автономной операционной системы (ОС) и рабочих программ. В настоящее время такой подход почти не используется, потому что требует больших трудозатрат;

• системы, использующие стандартное математическое обеспечение ЭВМ. Такие системы ориентированы из структуру информационного фонда, которую поддерживает ОС. Известно, что основой таких структур является файл.

В настоящее время наряду с понятием "информационная система" используется и понятие "банк данных" (БД). Мы рассматриваем БД как разновидность информационной системы.

2. ТЕХНОЛОГИЯ ПОСТРОЕНИЯ БАЗ ДАННЫХ

Основой современной информационной технологии по праву называют концепцию баз данных. Понятие "информационная технология" включает в себя устройство обработки и носитель информации, методы хранения переработки и обмена информации Разработка информационных систем на основе баз данных - процесс сложный, требующий соответствующей квалификации.

Исторические предпосылки Информационные системы (поисковые, документальные, фактографические) разрабатываются и используются давно с появлением первых ЭВМ. Первые системы разрабатывались на основе так называемого позадачного метода, т.е. они были специализированными для конкретных предметных областей (задач). Опыт и разработка сложных систем показали, что такой (позадачный метод) подход имеет существенные недостатки:

- возникла проблема контроля избыточности данных. Для различных задач (например, зарплата, кадры, производство), которые имеют одинаковые данные, данные дублировались. Это приводило к изменению всех данных при изменении их в одной задаче;

- наметилась взаимосвязь между данными и прикладными программами. Во-первых, прикладные программы имели описание данных и алгоритмы их обработки. Во-вторых, любые изменения в описании данных приводили к изменению программы. В-третьих, в программах дублировались идентичные алгоритмы;

- появитесь проблемы автоматизации создания и обработки информационного кодов, разделения логического и физического описания и обработки.

Появление операционных систем, в какой то мере решило указанные проблемы. Знаем, что одной из основных функций ОС - это управление данными. Система управления данными ОС (СУД ОС) обеспечивает пересылку данных между ОП и ВЗУ. Основные функции СУД ОС:

- присвоение определенной структуры данным, их каталогизация, размещение, запоминание, поиск и обновление.

Если рассмотреть, например ОС ЕС, то для связи с управляющей программой предусмотрен набор макрокоманд ввода-вывода (33 основных макрокоманд), действие которых зависит от метода доступа к данным. Имеется 9 методов доступа, каждый из которых характеризуется определенным набором макрокоманд СУД и требует использования определенных типов записей и их форматов.

С другой стороны, выбор типа формата записи зависит от информации, типа устройства, на котором она будет храниться, метода доступа, который будет использован при чтении и записи данных. Отметим также, что каждый из операторов *языка управления заданиями* (ЯУЗ) имеет несколько операндов (например, для оператора DD предусмотрено 15 операндов).

В среде ОС ЕС имеются также трудности при организации личных библиотек программ, описании сложных (сетевых) структур данных и их обработки. Любое изменение в структуре хранения приводит к модификации и перетрансляции обрабатывающих программ.

Эти и другие факторы показывают, что ОС имеют ряд недостатков и неудобств, которые явно всплывают на поверхность при организации и обработке сложных структур данных, а создание локальных массивов приводит к значительному дублированию информации. Для адекватного отображения реального мира, информационная база должна представлять собой единое взаимоувязанное целое Функции создания к ведения информационного фонда являются общими для различных категорий пользователей. Их можно отделить от других функций и привязать к специальному программному обеспечению.

Обобщение опыта разработки и использования информационных систем привело к выявлению их общих черт и пониманию того, что они должны строиться с собственной базой данных с использованием некоторого аппарата, получившею название системы правления базами данных (СУБД).

Базы и банки данных. Информационная система хранить и обрабатывает совокупность элементов данных, которые являются отображением некоторой реальной предметной области. Для выявления данных, подлежащих хранению и последующей обработке, предметная область разбивается на отдельные единицы с указанием связей между ними.

Базой данных называется поименованная структурированная совокупность взаимосвязанных данных конкретной предметной области, находящихся под централизованным программным управлением [2].

Как отмечается в научной литературе, термин "банк данных" возник в конце 60-д годов. Первоначально под *банком данных* подразумевали совокупность взаимосвязанных файлов, находящихся под общим управлением. Позднее в этом смысле стали использовать термин "базы данных". В настоящее время под банком данных понимается автоматизированная система языковых, программных, технических средств организации, накопления и коллективного использования данных. Необходимыми компонентами БД являются базы данных и одна или несколько СУБД.

Требования к базам данных. Современные информационные системы разрабатываются на основе *технологии баз данных.* Особенности такой технологии и требования к базам данных следующие:

1. Наличие единого целостного отображения предметной области. Это сокращает избыточность хранимых данных и дает возможность простого контроля целостности данных.

2. При проектировании конкретных приложений, программисты освобождаются от многих повторяемых функций по организации данных.

3. Наличие и использование отдельных языковых и программных средств делает базу данных доступной как для программистов, так и для конечных пользователей.

4. Такой подход обеспечивает максимально возможную независимость прикладных программ от данных, т.е. отделение логической модели предметной области от физического представления в памяти ЭВМ.

5. Пользователи имеют простой доступ к данным. Сложный доступ осуществляет сама СУБД. Они могут легко узнать, какие данные имеются в их распоряжении.

6. Такая технология расширяет возможности СУД ОС и дает больше, чем обычная файловая система.

Система управления базами данных. Мы отметили, что будем рассматривать такие информационные системы, которые разрабатываются на основе СУБД. *Система управления базами данных* - это совокупность программных и языковых средств для создания баз данных, поддержки их актуальности и организации к ним доступа различных пользователей.

Главная задача СУБД заключается в обеспечении пользователя инструментарием, позволяющим оперировать данными в абстрактных терминах, не снизанных со способами их хранения в ЭВМ. СУБД должна выполнять также и ряд следующих функций [3]:

* Обеспечение секретности. Не каждый пользователь должен иметь доступ к данным;

- Защита целостности данных. СУБД может проверять некоторого рода ограничения непротиворечивости данных, если ей это предписано;

- Синхронизация. Когда исполняется несколько программ, одновременно осуществляющих доступ к базе данных, то СУБД должна обеспечивать защиту от нарушения операций над элементом данных;

- Защита от отказов и восстановление. Должны быть предусмотрены средства, обеспечивающие регулярное создание копий базы данных для ее восстановления после устранения сбоев оборудования или ошибок программного обеспечения.

3. ОПИСАНИЕ АРХИТЕКТУРЫ СУБД

Общая структура (архитектура) СУБД содержит четыре основных уровня - уровня физической модели, логической модели данных, внешних моделей и прикладной уровень. Эти уровни показаны на следующем рисунке [4].

Физическая модель данных (внутренняя модель данных, общая физическая схема) располагается на самом нижнем уровне в структуре СУБД Это машинная схема базы данных Она содержит сведения о характеристиках устройств внешней памяти, форматах записей, индексах.

Логическая модель данных (ЛМД) - абстрактное представление БД в терминах логически* характеристик данных и отношений между элементами данных (поля, записи, отношения). Другое название логической модели - *концептуальная модель данных* (общая логическая схема БД**).** Логическая модель выражается в СУБД с помощью концептуальной схемы, или просто схемы.

Схема описывается на специальном языке. Такие схемы имеют определенную структуру, т.е. описываются определенной моделью.

Внешняя модель данных –Это часть логической модели, которая «невидима» определенному пользователю. Могут быть разные внешние модели, представленные в виде подсхем конкретных пользователей.

Прикладной уровень СУБД охватывает прикладные программы, предназначенные для выполнения в среде СУБД. Поэтому прикладные программы разрабатываются пользователями и не входят в состав системного программного обеспечения СУБД. Для написания прикладных программ могут быть использованы традиционные языки программирования (Фортран, Кобол, ПЛ/1, Ассемблер и др.) и специальные языки.

Языковые и программные средства СУБД. Для организации и обработки БД СУБД предоставляет языковые и программные средства. Языки используются для описания структуры хранения БД и манипулирования данными. Основными языками являются *язык описания* (определения) данных (ЯОД) и *язык манипулирования данными* (ЯМД).

ЯОД предназначен для определения логической модели БД. Именно средствами ЯОД описываются общая схема и подсхемы пользователей. Общую логическую схему формирует администратор (разработчик) информационной системы. ЯОД состоит из множества средств (конструкции, операторов) описания данных, которые очень близки к средствам описания данных в языках программирования . Схемы, подсхемы, схемы хранения проектируются и описываются на ЯОД. Эти описания переносятся на машинные носители, переводятся на объектные или загрузочные представления.

ЯМД используется для обращения к БД из программ, написанных на обычных языках (Фортран, Кобол, ПЛ/1, Паскаль) программирования. Когда на языке манипулирования данными запрашивается некоторая запись БД, эта запись считывается на рабочую область прикладной программы и там

обрабатывается. Или же, когда запись необходимо включить в БД прикладная программа помещает ее в рабочую область и ЯМД включает ее в БД.

Кроме указанных языков СУБД может иметь и другие языки, такие как языки ведения диалога, распределения ресурсов, описания сценариев и т.п.

Программные средства (системные программы) СУБД представляют сложный комплекс: программа управления данными, трансляторы с языков (ЯОД, ЯМД) и утилиты взаимодействия пользователей и технических средств. Все программы СУБД выполняются непосредственно под управлением ОС. Мы говорили, что СУБД это сложные системы, которые объединяют разнотипные компоненты и выполняют различные функции. Классификация СУБД производится по разным аспектам. Если рассмотреть по используемому языку общения (ЯМД), то СУБД можно типизировать на СУБД с замкнутыми языками и СУБД с базовыми языками, а также на системы программирования баз данных (СПБД) на основе языка программирования баз данных (ЯПБД).

Замкнутые СУБД имеют собственные самостоятельные языки общения пользователя с БД. Такие системы "удобны" пользователям-непрограммистам, которые работают с системой с помощью готовых сценариев в режиме "вопрос-ответ". Возможности замкнутых систем зависят от набора встроенных функций высокого уровня, которые близки к форме записи на обычном естественном языке.

СУБД с базовыми языками для обращения и обработки БД используют обычные языки программирования (Кобол, Фортран, ПЛ/1, Ассемблер...), "расширенный" операторами ЯМД.

Существует несколько подходов применения ЯМД в программах обработки данных. Первый из них основан на использовании макрокоманды CALL которая вызывает нужную подпрограмму оператора ЯМД.

После выполнения оператора ЯМД управление передается программе пользователи. Второй подход предусматривает включение операторов ЯМД в программу на базовом языке. На этапе пред трансляции операторы ЯМД заменяются операторами базового языка, после чего осуществляется трансляция программы с базового языка, т.е. выполняется двухэтажная трансляция программы. Использование СУБД с базовыми языками требует знания программирования и программиста для общения с БД. С другой стороны, на таких языках трудно реализовать нерегламентированные запросы к системе.

Эти недостатки замкнутых систем и СУБД с базовыми языками в какой-то мере устраняют СПБД. Суть применения СПБД заключается в том, что язык программирования включает в себя средства конструирования и обработки баз данных. Характерным признаком СПБД является компиляционный способ исполнения заданий (обычно многие СУБД - интерпретирующие системы). При этом любое задание к системе рассматривается как текст на входном языке - языке программирования БД.

Этапы создания ИС с базами данных на основе СУБД. Основные цели разработки информационных систем заключаются в том, что предоставить пользователю необходимые данные за приемлемые сроки, а также

удовлетворить изменчивые е! о требования. Так как система баз данных состоит из данных и программ, то проектирование и реализация ИС с базой данных являются частью общей методологии создания систем программного обеспечения.

В настоящее время существуют различные методы проектирования ИС и развитие их методологии продолжается. На следующем рисунке показаны взаимосвязь и итеративный характер процесса разработки ИС с базой данных [5].

На первом этапе строится предварительная логическая структура, которая отображает предметную область и требования пользователя системы. Предметная область описывается независимыми от конкретной СУБД приемами и терминами, т.е. выполняется этап информационно-логического (мифологического) описания предметной области и создается *инфологическая модель* (ИЛМ) предметной области. Такая модель должна обеспечивать адекватное отображение предметной области и быть динамичной.

Модель данных логического уровня, поддерживаемую средствами СУБД называют *даталогической моделью* базы данных (ДЛМ). Эта модель представляет собой отображение логических связей между элементами данных в терминах и ограничениях конкретной СУБД. Мифологическая модель предметной области является исходной по отношению к даталогической модели БД. Модель данных физического уровня *(физическая моде ъ данных* - ФМД) связывает даталогическую модель БД со средой хранения.

Из рисунка видно, что в процессе анализа запросов пользователей (требования, пожелания, дополнения) даталогическая и физическая модели данных могут быть перестроены или перепроектированы. Таким образом, шаг за шагом, итеративно разрабатывайся и развивается версия информационной системы.

4. ПОСТРОЕНИЕ МОДЕЛИ ДАННЫХ

Выше было сказано, что в информационных системах отображение предметной области представляется моделями нескольких уровней, таких как инфологическая, даталогическая и физическая.

Современные СУБД на даталогическом уровне поддерживают разные модели данных. Наиболее изучены и распространены иерархическая, сетевая и реляционная модели данных. Простыми словами, *модель данных* - это общая схема, изображающая структуру связи между совокупностями элементов данных предметной области (базы данных).

Более строгие определение понятия модели данных дал Э. Кодд в своей работе [7], Он выделяет в модели данных три наиболее существенные компоненты:

- совокупность средств определения допустимых структур данных;
- множество операции, применяемых к допустимому состоянию базы данных для поиска или модификации данных;
- множество ограничений целостности, явно или неявно определяющих множестве допустимых состояний базы данных.

Каждая база данных создается в соответствии с той или иной моделью данных. Поэтому говорят, что СУБД поддерживает ту или иную модель данных. Существую также такие СУБД, которые сочетают в себе особенности разных моделей и дают возможность описания и использования нескольких моделей. Такие системы называются *"смешанными"* или *"мультимоделъными"* системами. В качестве примера отметим, что иерархическая модель воплощена в системах ИНЕС [8] и 1М8 [6]. Сетевой подход был описан в работе Ч. Бахмана и группы КОДАСИЛ (Ассоциация по языкам течем обработки данных). Примерами систем такого рода являются "Банк-ОС" [9] и "Сетор" [10]. Реляционный подход был предложен в работах Э. Кодда Щ]. Примерами реляционных систем являются Пальма, Р4ОКЕ5, семейства dBase, Clipper. Системы типа БОЛЗ-6 и Атлант [12] можно отнести к группе мультимодельных систем.

Иерархическая модель данных. Система ИНЕС. Учитывая широкое распространение системы ИНЕС рассмотрим ее возможности для организации иерархических баз данных по работам [8,5]. Отметим, что многие существующие СУБД основаны на иерархическом подходе. В таких системах для создания логической базы данных (ЛБД) приняты следующие понятия.

Поле - минимальный поименованный элемент данных. *Сегмент* - поимениванная единица данных фиксированной длины, которая может содержать одно или несколько полей. Совокупность сегментов составляет *запись логической базы данных,* а совокупность записей составляет *логическую базу данных.* Основным способом связи сегментов является древовидная (иерархическая) структура, которая и служит для образования записи базы данных. Очевидно, что в иерархически связанных сегментах одного дерева допускается не более одной связи между сегментами, по этому она не

именуется. Связь между различными деревьями организуется с помощью *сегментов-указателей,*

Система ИНЕС оперирует с большим набором типов логических единиц данных. Их можно разделить на три группы :структурный, ссылочный и элементарный. В системе используется два вида деревьев. Дерево описания данных (ДОД), которое содержит графическое отображение даталогической модели и дерева данных (ДД), которое соответствует базе данных. Структурный тип соответствует вершине ДУД и может быть в двух типах; структура и массив.

Структура делится на простую и условную. Простая структура представляет собой тип данных, состав которых один и тот же независимо от их значений. Условная структура является типом данных, в котором состав данных может меняться в зависимости от их значений.

Другим видом структурного типа является массив. Тип массива имеет подчиненную вершину. Массив бывает простым, нумерованным и ключевым. В простом массиве данные хранятся и обрабатываются последовательно. В нумерованном массиве каждому элементу присваивается номер. Нумерация задается при создании массива. Для каждого элемента ключевого массива выделяется вершина, которая является ключом. По ключу выполняется произвольный доступ к элементу.

5. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ИНТЕГРАЛЬНОЙ СИСТЕМЫ «DELPHI – 6 »

Delphi - это греческий город, где жил дельфийский оракул. И этим именем был назван новый программный продукт с феноменальными характеристиками.

Delphi - это комбинация нескольких важнейших технологий:

- Высокопроизводительный компилятор в машинный код
- Объектно-ориентированная модель компонент
- Визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов
- Масштабируемые средства для построения баз данных

Компилятор в машинный код

Компилятор, встроенный в **Delphi**, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре “клиент-сервер”. Этот компилятор в настоящее время является самым быстрым в мире, его скорость компиляции составляет свыше 200 тысяч строк в минуту на компьютере Pentium 3. Он предлагает легкость разработки и быстрое время проверки готового программного блока, характерного для языков четвертого поколения (4GL) и в то же время обеспечивает качество кода, характерного для компилятора 3GL. В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты как художник, делающий крупные мазки кистью. Еще до компиляции он видит результаты своей работы - после подключения к источнику данных их можно видеть отображенными на форме, можно перемещаться по данным, представлять их в том или ином виде. В этом смысле проектирование в **Delphi** мало чем отличается от проектирования в интерпретирующей среде, однако после выполнения компиляции мы получаем код, который исполняется в 20-30 раз быстрее, чем то же самое, сделанное при помощи интерпретатора. Кроме того, компилятор компилятору рознь, в **Delphi** компиляция производится непосредственно в родной машинный код, в то время как существуют компиляторы, превращающие программу в так называемый p-код, который затем интерпретируется виртуальной p-машиной. Это не может не сказаться на фактическом быстродействии готового приложения.

Объектно-ориентированная модель программных компонент

Основной упор этой модели в **Delphi** делается на максимальном реиспользовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а также дает им возможность создавать свои собственные объекты для среды **Delphi**. Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует. Действительно, все в **Delphi** написано на нем же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки. В результате нет никакой

разницы между объектами, поставляемыми Borland или третьими фирмами, и объектами, которые вы можете создать.

Быстрая разработка работающего приложения из прототипов. Среда **Delphi** включает в себя полный набор визуальных инструментов для скоростной разработки приложений (RAD - rapid application development), поддерживающей разработку пользовательского интерфейса и подключение к корпоративным базам данных. VCL - библиотека визуальных компонент, включает в себя стандартные объекты построения пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами, управление DDE и OLE. Единственное, что можно поставить в вину **Delphi**, это то, что готовых компонент, поставляемых Borland, могло бы быть и больше. Однако, разработки других фирм, а также свободно распространяемые программистами freeware-компоненты уже восполнили этот недостаток. Соответствующий стандарт компонент назывался VBX. И этот стандарт так же поддерживается в **Delphi**. Однако, визуальные компоненты в **Delphi** обладают большей гибкостью.

Масштабируемые средства для построения баз данных. Объекты БД в **Delphi** основаны на SQL и включают в себя полную мощь Borland Database Engine. В состав **Delphi** также включен Borland SQL Link, поэтому доступ к СУБД Oracle, Sybase, Informix и InterBase происходит с высокой эффективностью. Кроме того, **Delphi** включает в себя локальный сервер Interbase для того, чтобы можно было разработать расширяемые на любые внешние SQL-сервера приложения в офлайновом режиме. Разработчик в среде **Delphi**, проектирующий информационную систему для локальной машины (к примеру, небольшую систему учета медицинских карточек для одного компьютера), может использовать для хранения информации файлы формата **.dbf** (как в dBase или Clipper) или **.db** (Paradox). Если же он будет использовать локальный InterBase for Windows 4.0 (это локальный SQL-сервер, входящий в поставку), то его приложение безо всяких изменений будет работать и в составе большой системы с архитектурой клиент-сервер.

Delphi for Windows. Delphi for Windows представляет из себя подмножество **Delphi** Client-Server и предназначен для разработчиков высокопроизводительных персональных приложений, работающих с локальными СУБД типа dBase и Paradox.**Delphi** Desktop Edition предлагает такую же среду для быстрой разработки и первоклассный компилятор как и клиент-серверная версия (Client/Server Edition). Эта среда позволяет разработчику быстро изготавливать персональные приложения, работающие с персональными СУБД типа dBase и Paradox. **Delphi** позволяет также создавать разработчику DLL, которая может быть вызвана из Paradox, dBase, C++ или каких-нибудь других готовых программ.

В Delphi for Windows, как и в Delphi Client-Server, входят

- компилятор Object Pascal (этот язык является расширением языка Borland Pascal 7.0)

- генератор отчетов ReportSmith 2.5 (у которого, правда, отсутствует возможность работы с SQL-серверами)
- среда визуального построителя приложений
- библиотека визуальных компонент
- Локальный сервер InterBase

RAD Pack for Delphi. В RAD Pack for **Delphi** входит набор полезных дополнений, которые помогут разработчику при освоении и использовании **Delphi**. Это учебник по объектному паскалю, интерактивный отладчик самой последней версии, Borland Visual Solutions Pack (набор VBX для реализации редакторов, электронных таблиц, коммуникационные VBX, VBX с деловой графикой и т.п.), Resource WorkShop для работы с ресурсами Borland Pascal 7.0, а также дельфийский эксперт для преобразования ресурсов BP 7.0 в формы **Delphi**.

В первую очередь **Delphi** предназначен для профессионалов-разработчиков корпоративных информационных систем. Не секрет, что некоторые удачные продукты, предназначенные для скоростной разработки приложений (*RAD - rapid application development*) прекрасно работают при изготовлении достаточно простых приложений, однако, разработчик сталкивается с непредвиденными сложностями, когда пытается сделать что-то действительно сложное. Бывает, что в продукте вскрываются присущие ему ограничения только по прошествии некоторого времени. **Delphi** такие ограничения не присущи. Хорошее доказательство тому - это тот факт, что сам **Delphi** разработан на **Delphi**. Однако **Delphi** предназначен не только для программистов-профессионалов.

Руководители предприятий, планирующие выделение средств на приобретение программных продуктов, должны быть уверены в том, что планируемые инвестиции окупятся. Поэтому одним из оцениваемых факторов должен быть вопрос - а легко ли найти специалиста по **Delphi** и сколько будет стоить его обучение, сколько времени специалист затратит на овладение продуктом. Ответ здесь получить весьма просто - любой программист на Pascal способен практически сразу профессионально освоить **Delphi**. Специалисту, ранее использовавшему другие программные продукты, придется труднее, однако самое первое работающее приложение он сможет написать в течение первого же часа работы на **Delphi**. И, конечно же, открытая технология **Delphi** является мощным гарантом того, что инвестици, сделанные в **Delphi**, будут сохранены в течение многих лет.

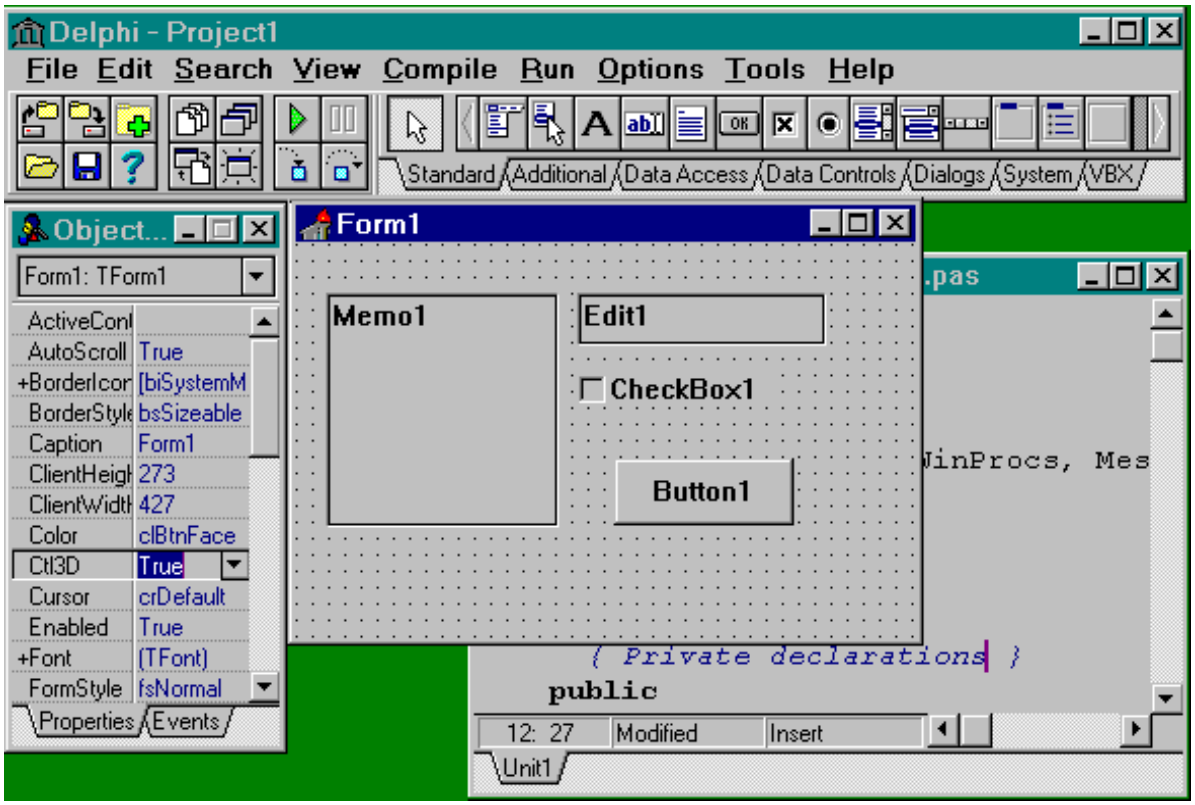
Некоторые особенности Delphi .*Локальный сервер InterBase* - следует заметить, что этот инструмент предназначен только для автономной отладки приложений. В действительности он представляет из себя сокращенный вариант обработчика SQL-запросов InterBase, в который не включены некоторые возможности настоящего сервера InterBase. Отсутствие этих возможностей с лихвой компенсируется преимуществом автономной отладки программ.

Team Development Support - средство поддержки разработки проекта в группе. Позволяет существенно облегчить управление крупными проектами. Это сделано в виде возможности подключения такого продукта как Intersolve PVCS 5.1 непосредственно к среде **Delphi.**

Высокопроизводительный компилятор в машинный код - в отличие от большинства Pascal - компиляторов, транслирующих в р-код, в **Delphi** программный текст компилируется непосредственно в машинный код, в результате чего **Delphi**- приложения исполняются в 20-30 раз быстрее (особенно приложения, использующие математические функции). Готовое приложение может быть изготовлено либо в виде исполняемого модуля, либо в виде динамической библиотеки, которую можно использовать в приложениях, написанных на других языках программирования.

Поддержка OLE 2.0, DDE и VBX Это очень важная особенность для разработчиков в среде Windows, поскольку в уже существующие Windows-приложения программист может интегрировать то, что разработает при помощи **Delphi**.

Delphi: настраиваемая среда разработчика



После запуска **Delphi** в верхнем окне горизонтально располагаются иконки палитры компонент. Если курсор задерживается на одной из иконок, под ней в желтом прямоугольнике появляется подсказка

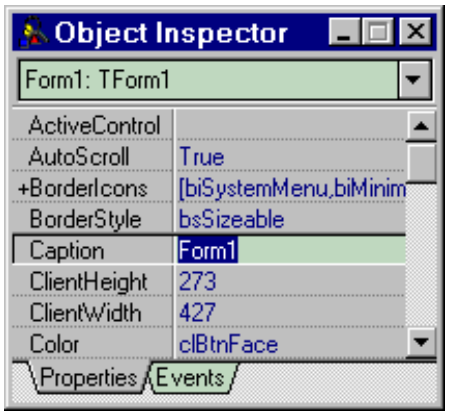
Из этой палитры компонент вы можете выбирать компоненты, из которых можно строить приложения. Компоненты включают в себя как визуальные, так и логические компоненты. Такие вещи, как кнопки, поля редактирования - это визуальные компоненты; а таблицы, отчеты - это логические.

Понятно, что поскольку в **Delphi** вы визуальным образом строите свою программу, все эти компоненты имеют свое графическое представление в поле форм для того, чтобы можно было бы ими соответствующим образом оперировать. Но для работающей программы видимыми остаются только визуальные компоненты. Компоненты сгруппированы на страницах палитры по своим функциям.

Delphi позволяет разработчикам настроить среду для максимального удобства. Вы можете легко изменить палитру компонент, инструментальную линейку, а также настраивать выделение синтаксиса цветом.

Заметим, что в **Delphi** вы можете определить свою группу компонент и разместить ее на странице палитры, а если возникнет необходимость, перегруппировать компоненты или удалить неиспользуемые.

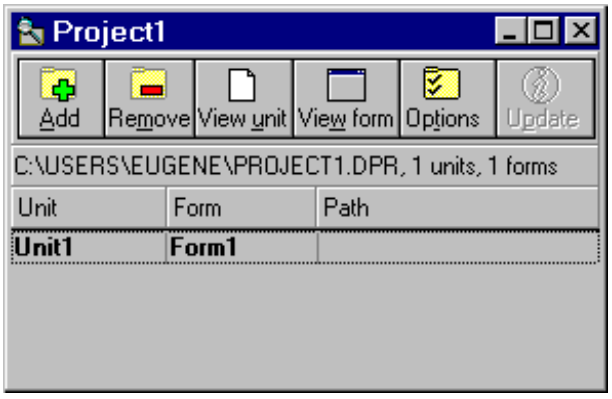
Инспектор объектов



Этот инструмент представляет из себя отдельное окно, где вы можете в период проектирования программы устанавливать значения свойств и событий объектов (Properties & Events).

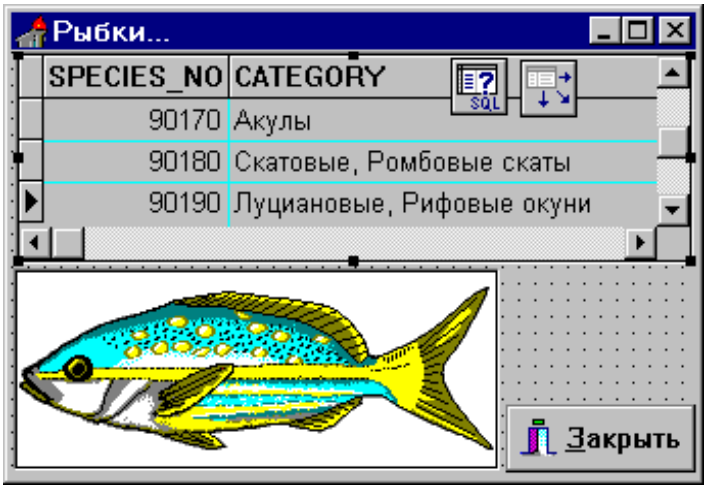
Менеджер проектов.

Дает возможность разработчику просмотреть все модули в соответствующем проекте и снабжает удобным механизмом для управления проектами.



Менеджер проектов показывает имена файлов, время/дату выбранных форм и пр.Можно немедленно попасть в текст или форму, просто щелкнув мышкой на соответствующее имя.

Разработка приложений БД



Delphi позволяет использовать библиотеку визуальных компонент для быстрого создания надежных приложений, которые легко расширяются до приложений с архитектурой клиент-сервер. Другими словами, Вы можете создать приложение, работающее с локальным сервером InterBase, а затем использовать созданное приложение, соединяясь с удаленным SQL-сервером через SQL-Links.

Внешний вид среды программирования Delphi отличается от многих других из тех, что можно увидеть в Windows. К примеру, Borland Pascal for Windows 7.0, Borland C++ 4.0, Word for Windows, Program Manager - это все MDI приложения и выглядят по-другому, чем Delphi. MDI (Multiple Document Interface) - определяет особый способ управления нескольких дочерних окон внутри одного большого окна.

Главные составные части среды программирования

Ниже перечислены основные составные части Delphi:

1. Дизайнер Форм (Form Designer)
2. Окно Редактора Исходного Текста (Editor Window)
3. Палитра Компонент (Component Palette)
4. Инспектор Объектов (Object Inspector)
5. Справочник (On-line help)

Есть, конечно, и другие важные составляющие Delphi, вроде линейки инструментов, системного меню и многие другие, нужные Вам для точной настройки программы и среды программирования.

Программисты на Delphi проводят большинство времени переключаясь между Дизайнером Форм и Окном Редактора Исходного Текста (которое для краткости называют Редактор). Прежде чем Вы начнете, убедитесь, что можете распознать эти два важных элемента. Дизайнер Форм показан на рис.1, окно Редактора - на рис.2.

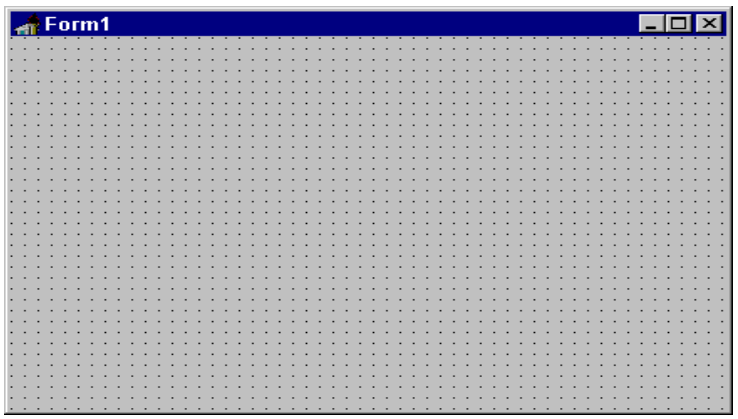


Рис.1: Дизайнер Форм - то место, где Вы создаете визуальный интерфейс программы.

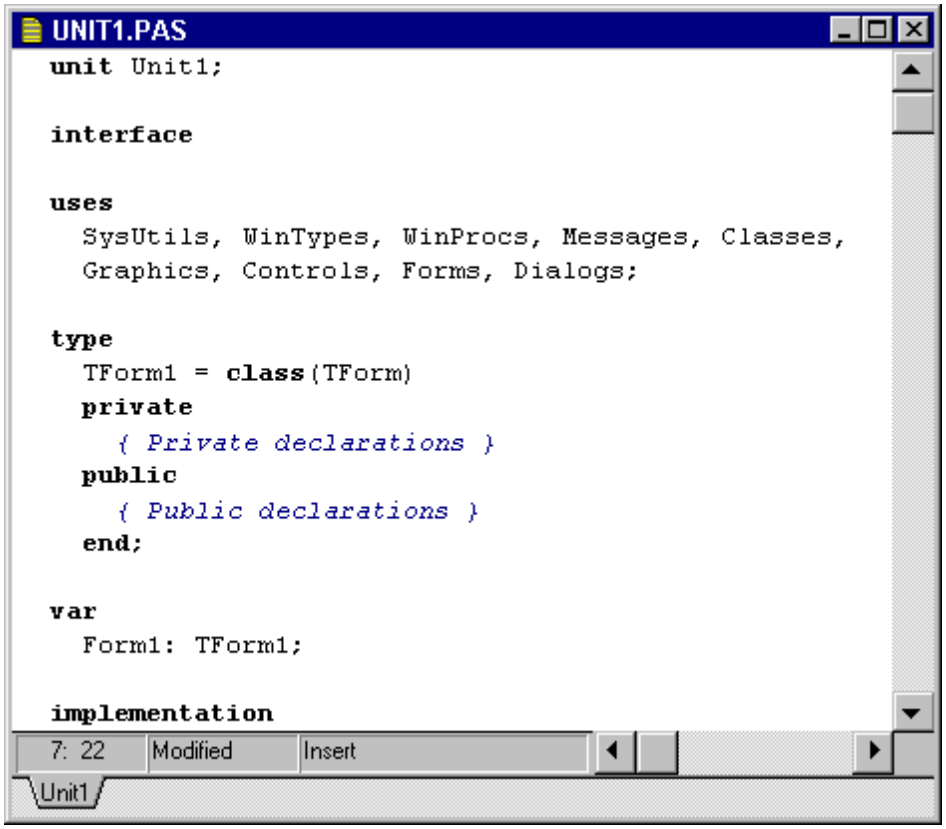


Рис.2: В окне Редактора Вы создаете логику управления программой.

Дизайнер Форм в Delphi столь интуитивно понятен и прост в использовании, что создание визуального интерфейса превращается в детскую игру. Дизайнер Форм первоначально состоит из одного пустого окна, которое Вы заполняете всевозможными объектами, выбранными на Палитре Компонент.

Несмотря на всю важность Дизайнера Форм, местом, где программисты проводят основное время является Редактор. Логика является движущей силой программы и Редактор - то место, где Вы ее “кодируете”.

Палитра Компонент (см. рис.3) позволяет Вам выбрать нужные объекты для размещения их на Дизайнере Форм. Для использования Палитры Компонент просто первый раз щелкните мышкой на один из объектов и потом второй раз - на Дизайнере Форм. Выбранный Вами объект появится на проектируемом окне и им можно манипулировать с помощью мыши.

Палитра Компонент использует постраничную группировку объектов. Внизу Палитры находится набор закладок - Standard, Additional, Dialogs и т.д. Если Вы щелкнете мышью на одну из закладок, то Вы можете перейти на следующую страницу Палитры Компонент. Принцип разбиения на страницы широко используется в среде программирования Delphi и его легко можно использовать в своей программе.

Рис.3: Палитра Компонент - место, где Вы выбираете объекты, которые будут помещены на вашу форму.

Предположим, Вы помещаете компонент TEdit на форму; Вы можете двигать его с места на место. Вы также можете использовать границу, прорисованную вокруг объекта для изменения его размеров. Большинством других компонент можно манипулировать тем же образом. Однако, невидимые во время выполнения программы компоненты (типа TMenu или TDataBase) не меняют своей формы.

Слева от Дизайнера Форм Вы можете видеть Инспектор Объектов (рис.4). Заметьте, что информация в Инспекторе Объектов меняется в зависимости от объекта, выбранного на форме. Важно понять, что каждый компонент является настоящим объектом и Вы можете менять его вид и поведение с помощью Инспектора Объектов.

Инспектор Объектов состоит из двух страниц, каждую из которых можно использовать для определения поведения данного компонента. Первая страница - это список свойств, вторая - список событий. Если нужно изменить что-нибудь, связанное с определенным компонентом, то Вы обычно делаете это в Инспекторе Объектов. К примеру, Вы можете изменить имя и размер компонента TLabel изменяя свойства Caption, Left, Top, Height, и Width.

Вы можете использовать закладки внизу Инспектора Объектов для переключения между страницами свойств и событий. Страница

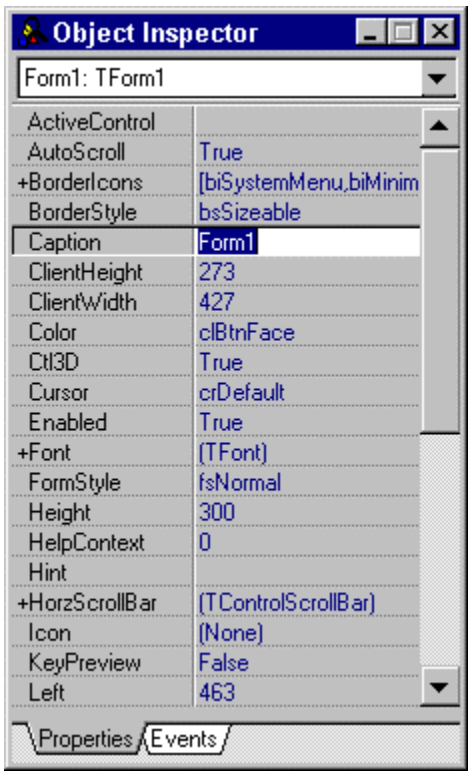


Рис.4: Инспектор Объектов позволяет определять свойства и поведение объектов, помещенных на форму.

событий связана с Редактором; если Вы дважды щелкнете мышкой на правую сторону какого-нибудь пункта, то соответствующий данному событию код автоматически запишется в Редактор, сам Редактор немедленно получит фокус, и Вы сразу же имеете возможность добавить код обработчика данного события.

Меню предоставляет быстрый и гибкий интерфейс к среде Delphi, потому что может управляться по набору “горячих клавиш”. Это удобно еще и потому, что здесь используются слова или короткие фразы, более точные и понятные, нежели иконки или пиктограммы. Вы можете использовать меню для выполнения широкого круга задач; скорее всего, для наиболее общих задач вроде открытия и закрытия файлов, управления отладчиком или настройкой среды программирования.

SpeedBar находится непосредственно под меню, слева от Палитры Компонент. SpeedBar выполняет много из того, что можно сделать через меню. Если задержать мышь над любой из иконок на SpeedBar, то Вы увидите что появится подсказка, объясняющая назначение данной иконки.

Инспектор Объектов

Основное для понимания Инспектора Объектов состоит в том, что он используется для изменения характеристик любого объекта, брошенного на форму. Кроме того, и для изменения свойств самой формы.

Лучший путь для изучения Инспектора объектов - поработать с ним. Для начала откройте новый проект, выбрав пункт меню File | New Project. Затем положите на форму объекты TMemo, TButton, и TListBox, как показано на рис.9.

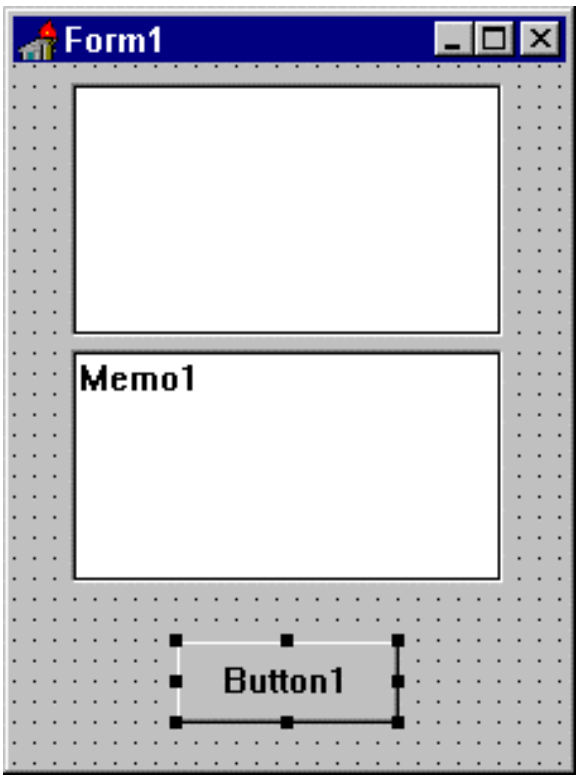


Рис.9: Простой объект TForm с компонентами TMemo, TButton, и TListBox.

6. ОБРАБОТКА ДАННЫХ В СТУДЕНЧЕСКОГО ОТДЕЛЕ

Студентский отдел СамГУ им. А. Навои имеет в вооружении качестве основного документа следующего листа:

Каждый студент заполняет эту таблицу, и эта таблица вводится в память компьютера. В дальнейшем этот документ является основной и в нем отражается все действие. В связи с этим был поставлен вопрос о том, что автоматизировать этот документ с использованием современных компьютерных технологий.

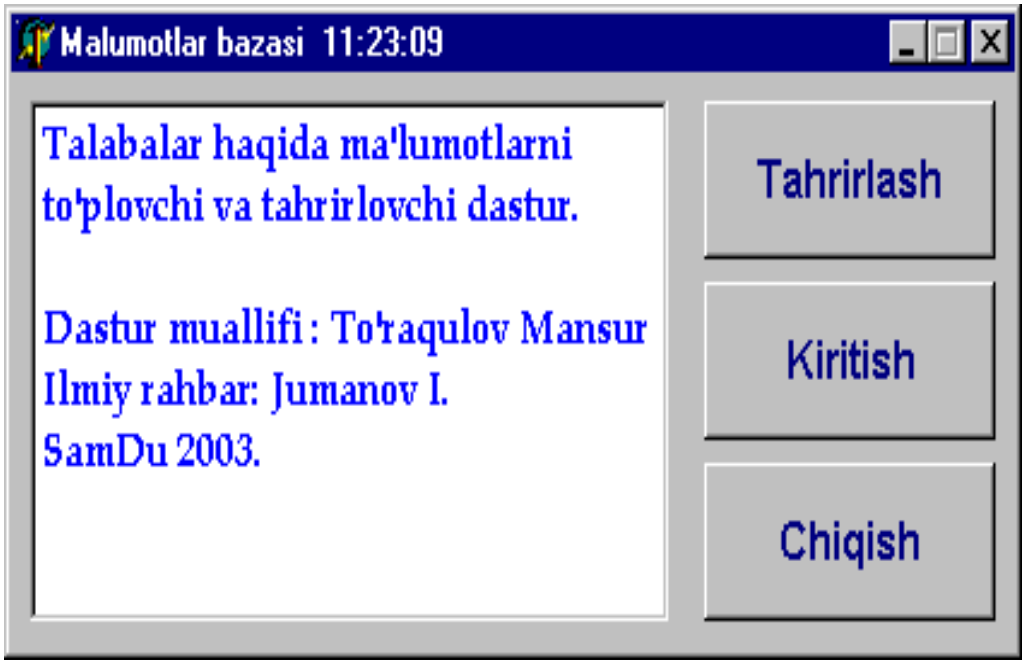
Решение вопроса.

Дипломная работа посявщена автоматизацию рабочих документов студенческого отдела СамГУ с использованием современных компьютерных технологий.

Дипломная работа оформлена в виде конечного продукта, т. е. в виде exe файла. Это означает, что инциализаация программ приводит действие системы. Сначала открывается анимационное окно с использованием структуры клипа окно следующего вида:

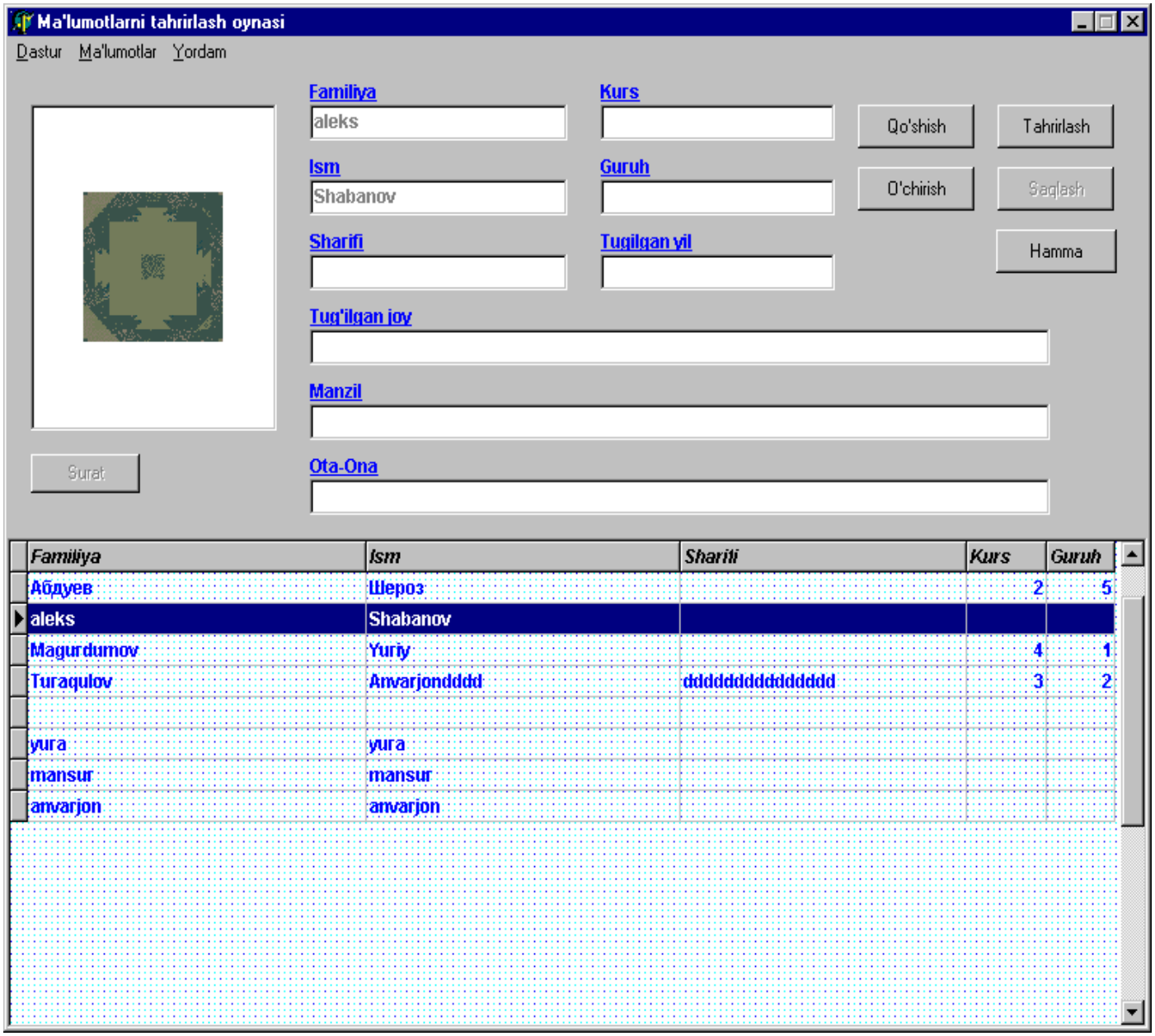
В нем есть сведения об автора и научного руководителя, и имеется три командная кнопка «Программа», «Дипломная работа» и «Выход». Главное окно красочно оформлено с использованием современных компьютерных технологий, таких как Photoshop 6, CorelDraw 8, Flash 5, Adobe Primer и т.д.

Приведя к действию командную кнопку «**Программа**» войдем в информационную систему «**Talaba**». Перед Вами открывается следующее диалоговое окно:



Оно снабжено командными кнопками «**Tahrirlash**» (редактирование), «**Kiritish**» (Добавление), «**Chiqish**» (Выход) и текстовым окном с содержанием о том, что программа предназначена для сбора и редактирования данных о студентах и имеется сведения об авторе и его научным руководителе.

Нажав не командную кнопку «**Tahrirlash** » открываем другое окно:



Окно имеем раздел меню, окна ввода, окно отображение базы данных, несколько командных кнопок и графическое окно.

В меню «**Dastur**» имеется подменю «**О программе**», которая дает сведения о программе.

Меню «**Ma'lumotlar**» (Данные) решает важную задачу. В нем существуют подменю «**Saralash**» (Сортировка) и «**Tartiblash**» (Упорядочивание).

Меню «**Yordam**» (Помощь) отвечает на некоторые вопросы пользователя, т.е. она снабжено справочным материалом о системе.

Окно ввода сведения о студентах состоит из нескольких пунктов, таких как:

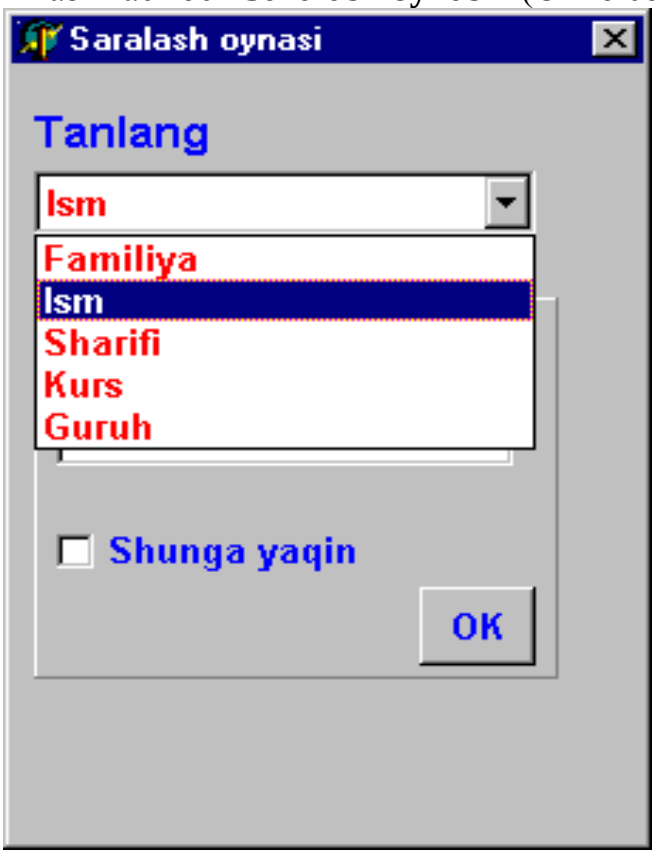
«Familiya» (Фамилия), «Ism» (Имя), «Sharif» (Отечество), «Kurs» (курс), «Guruh» (Группа), «Tugilgan yil» (Год рождения), «Tugilgan joy» (Место рождения), «Manzil» (Адрес), «Ota-Ona» (Сведения о родителях).

Внизу расположено окно отображения выделенных или всех студентов со всеми вышеуказанными сведениями.

Здесь командными кнопками являются «**Qo'shish**» (Добавление), «**O'chirish**» (Удаление), «**Tahrirlash**» (редактирование), «**Saqlash**» (Сохранение), «**Hamma**» (Все действия).

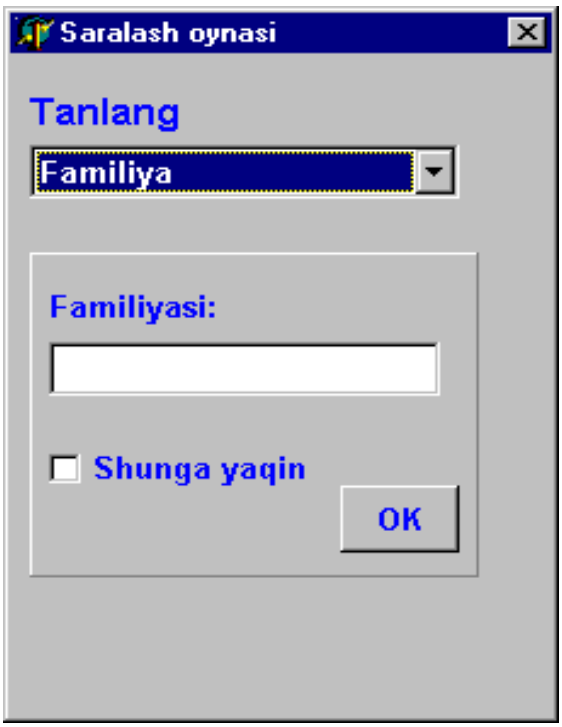
Графическое окно предназначено для отображения фотографии выбираемого студента. Для связывания базы данных и файла с фотокарточкой студента существует командная кнопка «**Surat**» (Фото). Нажатия, которой приводится в действие открывание диалогового окно поиска и выборки требуемого файла (файла с изображением фотокарточки).

Пользователь, выбрав подменю «**Saralash**» (Сортировка), который расположен в меню «**Ma'lumotlar**» (Данные) попадает в дополнительное диалоговое окно, так называемое «**Saralash oynasi**» (Окно сортировки).



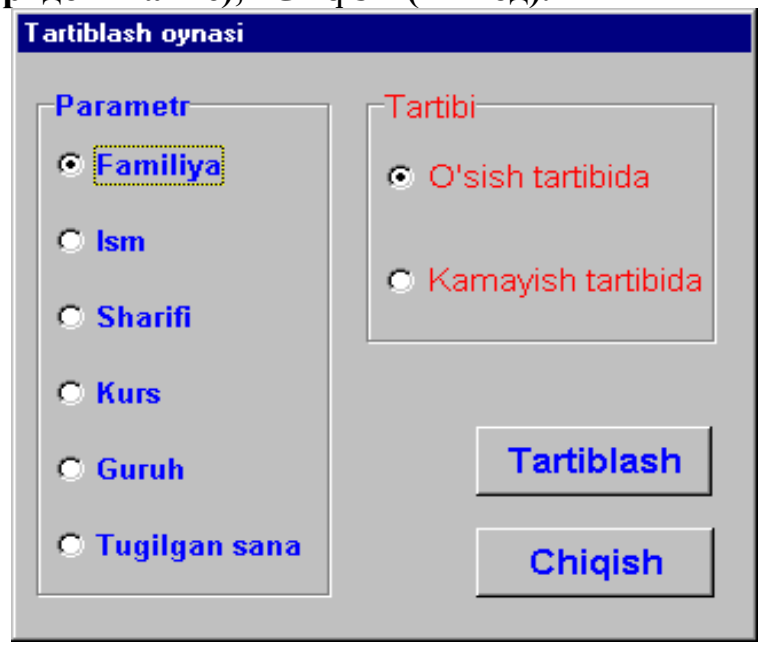
Здесь пользователь может производить сортировку по компонентам: «**Familiya**»; «**Ism**»; «**Sarifi**»; «**Kurs**»; «**Guruh**». Выбрав нужные компоненты и нажав на кнопку «ОК» пользователь возвращается в прежнее окно. Вводя соответствующие компоненты, он получает в нижнем окне полную информацию, которое хранится в БД системе.

В некоторых случаях, когда пользователь точно не знает значение выбираемого компонента, то он может воспользоваться дополнительный выборкой «**Shunga yaqin**» (Близко к этому). Например, пользователь, выбрав компонент «**Familiya**» воспользуется выборкой, то окно сортировка меняет свой вид и примет вид как в рисунке:



Здесь необходимо указать нижний и верхний предел номеров группы. В этом случае сортировка производится по указанным группам. Если пользователю необходимо произвести сортировку по фамилии, то здесь надо указать предельной значение первой буквы , например: от A до B или от L до P и т.д.

Меню ”Ma`lumotlar” **(Данные) имеет и подменю** “Tartiblash” **(упорядочивание). Выборов, которое открываем окно** “Tartiblash oynasi” **(окно упорядочивание). Оно состоит из двух фреймов: Parament (параметры) и Tartibi (порядок) и командных кнопок** “Tartiblash”**(упорядочивание),** ”Chiqish”**(Выход).**



Во фрейме **Paramentr** расположены такие параметры как: Familiya, Ism, Sharifi, Kurs, Guruh, Tugilgan sana, которые снабжены с перикаючателем. Выбрав нужный параметр, обходимо перейти к фрейму “**Tartibi**”. С помощью

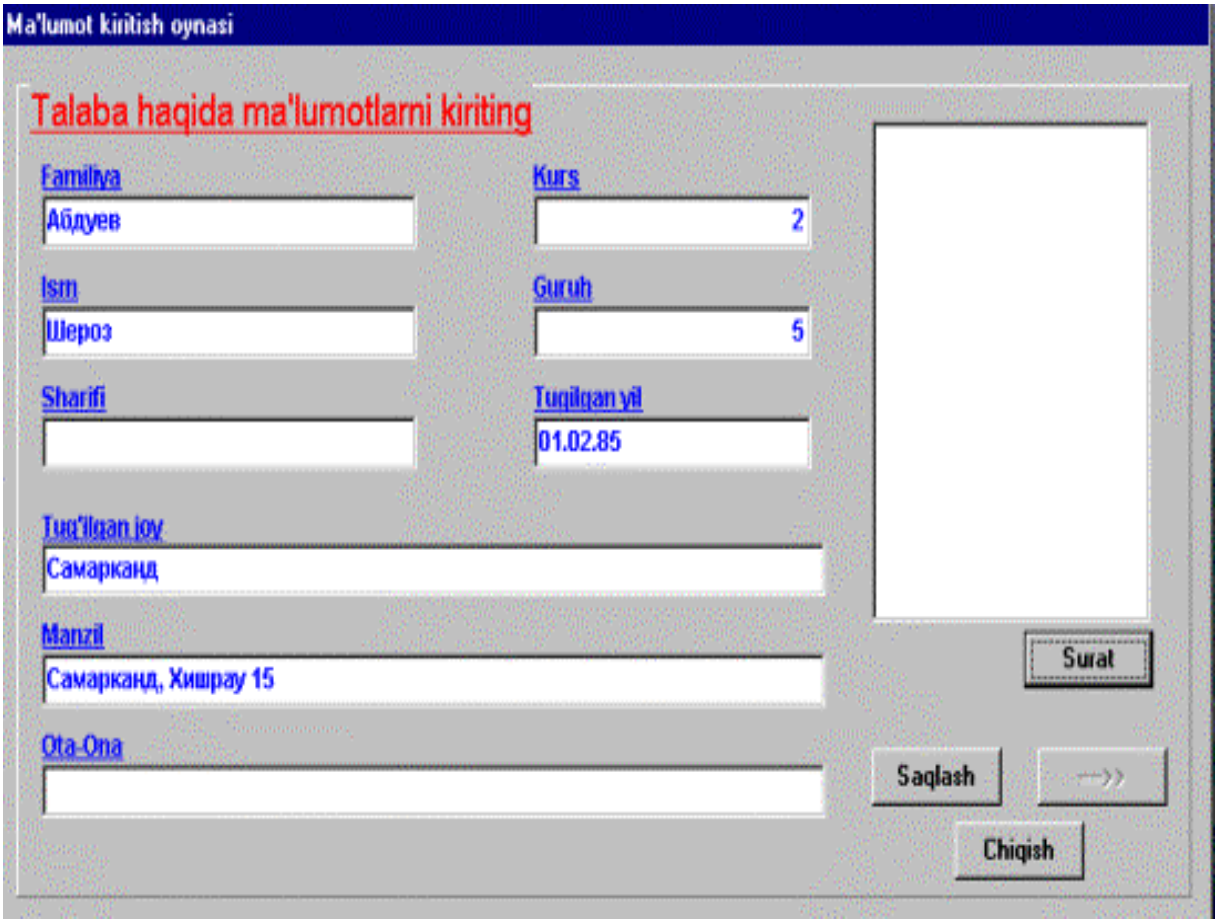
этого фрейма упорядочивание производится или “**O`shish tartibida**” (по увеличению) или «**Ramayich tartibida**» (по уменьшению).

Нажатие командной кнопки «**Tartiblash**» (упорядочивание) воспроизводит к упорядочиванию всех данных по выбранному методу.

Кнопка «**Chiqish**» (выход) соответствует к возвращению в прежнее диалоговое окно.

При необходимости добавления данных в БД системы надо воспользоваться командной кнопкой «**Kiritish**» (Ввод).

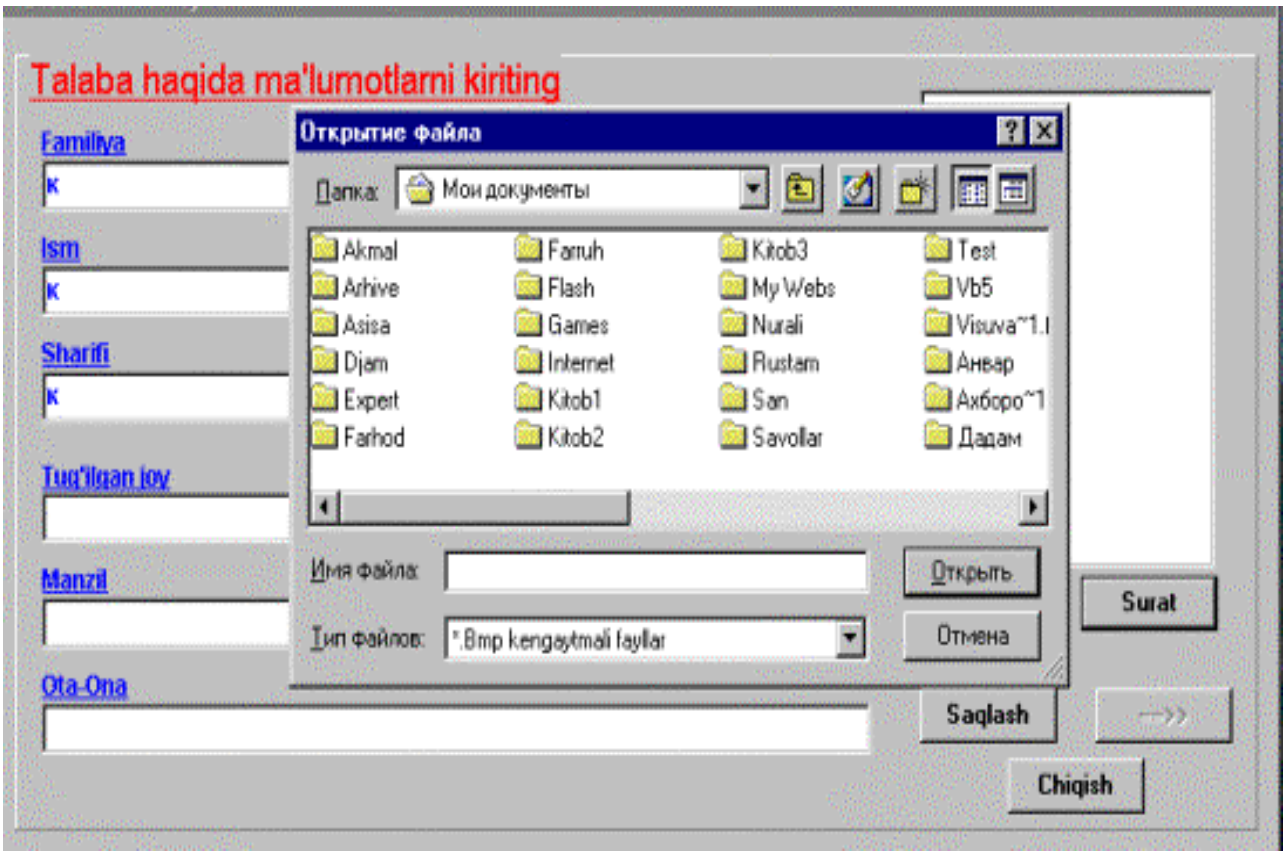
Оно открывает окно «**Ma’lumotlarni kiritish oynasi**» (Окно ввода информации) следующего вида:



Оно состоит из нескольких окон ввода информации с конкретными названиями, окно изображения и командных кнопок «**Surat**» (Фото), «**Saqlash**» (Сохранить), «→>» (Следующие) и «**Chiqish**» (Выход).

Окно «**Ma’lumotlarni kiritish oynasi**» (Окно ввода информации) имеет специальную специфику, по которой без заполнения некоторых окон ввода информации (например, Familiya»; «Ism»; «Sarifi»; «Kurs»; «Guruh») командная кнопка Saqlash» (Сохранить), «→» (Следующие) не активизируются.

Нажимав командную кнопку «Surat**» (Фото) получим диалоговое окно, из которой выбираем нужного файла с фотокарточкой.**


Talaba haqida ma'lumotlarni kiriting
Familiya
Ism
Sharifi
Tug'ilgan joy
Manzil
Ota-Ona
Открытие файла
Папка:
Мои документы
Akmal
Arhive
Asisa
Djam
Expert
Farhod
Faruh
Flash
Games
Internet
Kitob1
Kitob2
Kitob3
My Webs
Nurali
Rustam
San
Savollar
Test
Vb5
Анвар
Ахборо~1
Дадам
Имя файла:
Тип файлов:
*.Bmp kengaytmali fayllar
Открыть
Отмена
Surat
Saqlash
Chiqish

З А К Л Ю Ч Е Н И Е

1. Исследована структура информационных систем и технологии построения баз данных

2. Исследована архитектура СУБД и модели построения базы данных

3. Исследованы основные характеристики интегральной системы «delphi 6»

4. Разработаны модели и алгоритмы обработка данных в студенческого отделе

5. Разработаны программные средства информационной системы «ТАЛАБА» в ВУЗе и работоспособность их проверенный на реальных примерах