АННОТАЦИЯ

Пояснительная записка объемом 84 страницы, 3 приложения, 19 иллюстрации, 16 таблиц.

Автоматизация документооборота дает новые возможности любой организации по ускорению работы и оптимизации внутренних процессов. Автоматизация документооборота позволяет организациям существенно упростить проблемы, связанные с поиском, доступом и хранением документов, и как следствие избежать многих проблем, возникающих в процессе ведения документооборота.

В данном дипломном проекте разработана информационная система документооборота государственного бюджетного общеобразовательного учреждения города Москвы "Школа № 2095 "Покровский квартал". Проект разрабатывался на основе данных, собранных в период преддипломной практике по информационному, техническому, математическому и иным обеспечениям действующей системы документооборота учреждения.

Тема актуальна, так как решает вопрос повышения эффективности деятельности всего учебного заведения.

Проанализированы сущность задач документооборота в образовательном учреждении и исследована организация документооборота на примере учебного заведения; охарактеризовано входную, постоянную, промежуточную и исходную информацию и построено инфологическую модель системы.

На основе проведенного исследования разработано удобную в использовании и одновременно достаточно функциональную информационную систему "Документооборот" для операционной системы Windows®, которая автоматизирует учет документооборота в заведении и позволяет осуществлять выборку и печать отдельных документов.

СПИСОК ИСПОЛЬЗОВАННЫХ СОКРАЩЕНИЕ И ОБОЗНАЧЕНИЙ

АРМ – Автоматизированное рабочее место

АСУ – Автоматизированная система управления

ИС – Информационная система

КБД – Корпоративная база данных

ЛВС – Локальная вычислительная сеть

МТБ – Материально-техническая база

МН – Магнитный носитель

НД – Набор данных

НПД – Нормативно-правовые документы

ООП – Объектно-ориентированное программирование

ПО – Предметная область

ПК – Персональный компьютер

СУБД – Система управления базами данных

СУЭД – Система управления электронными документами

СЭД – Система электронного документооборота

ЭА – Электронные архивы

ЭД – Электронный документ

CRM – Customer Relation Management (Управления связями с клиентами)

СОДЕРЖАНИЕ

Введение 6
1. Аналитическая часть
1.1 Основные понятия документооборота 9
1.2 Классификация систем документооборота 13
1.3 Автоматизация документооборота в образовательном учреждении 20
1.4 Общая характеристика объекта исследования 26
1.5 Обзор существующих аналогов и разработок 27
1.6 Постановка задачи 30
2. Проектная часть
2.1 Обоснование выбора обеспечения реализации проекта 32
2.1.1 Информационное обеспечение 32
2.1.2 Выбор средств разработки 36
2.1.3 Технология доступа к данным 38
2.2 Проектирование БД 42
2.3 Разработка структуры программы 48
2.4 Разработка алгоритмов работы программы 51
2.5 Руководство пользователя программного обеспечения 57
3. Обоснование экономической эффективности проекта
3.1 Выбор и обоснование методики расчета 66
3.2. Расчет затрат и окупаемости реализации проекта 69
3.3 Расчет экономических результатов внедрения ПО 72
4. Экология и безопасность жизнедеятельности 74
Заключение 81
Список использованных источников 83
Приложения

ВВЕДЕНИЕ

Для любой современной организации важны возможности современных информационных технологий с точки зрения их использования в управлении документацией.

Когда речь заходит об автоматизации документооборота, то часто в эту сферу попадает множество программ, не имеющих прямого отношения к решаемой проблеме – текстовые редакторы, программы распознавания символов, системы электронной почты. Однако все это не является системой электронного документооборота. А учитывая нынешнее положение в мире, внедрение оптимально подходящей для организации системы документооборота, является чрезвычайно важной задачей.

Автоматизация документооборота дает новые возможности любой организации по ускорению работы и оптимизации внутренних процессов. Автоматизация документооборота позволяет организациям существенно упростить проблемы, связанные с поиском, доступом и хранением документов, и как следствие избежать многих проблем, возникающих в процессе ведения документооборота.

Система автоматизации документооборота достаточно сложный механизм и включает в себя множество подсистем, построенных с помощью программных различных продуктов.

Отсутствие оптимально структурированного документооборота и отсутствие возможности эффективно отслеживать движение и исполнение документов, приводит к снижению эффективности работы и увеличению ошибок.

Решение проблемы управления документацией в современных условиях позволит целенаправленно формировать информационные ресурсы организаций, сократить объем вводимой информации в ручную, что дает выигрыш во времени и обеспечить их эффективное функционирование.

Актуальность дипломной работы: Выбранная тема актуальна, поскольку известно, что организация работы с документами влияет на качество работы аппарата управления, организацию и культуру труда управленческих работников. От того, насколько профессионально ведется документация, зависит успех управленческой деятельности в целом. Как показывают современные исследования, 85% рабочего времени сотрудников организаций тратится на подготовку, сопровождение, заполнение, копирование и передачу документов.

Правильно организованное управление делами снижает время, необходимое для поиска, повышает точность и своевременность информации, устраняет ее избыточность. Рост объемов информации и, соответственно, документов, потребовал внедрения техники для своевременной обработки документов, а с появлением компьютеров – и самой информации.

Целью данного дипломного проекта является разработка системы электронного документооборота.

В качестве **поля исследования** выбрано государственное бюджетное общеобразовательное учреждение города Москвы "Школа № 2095 "Покровский квартал"

Для реализации поставленной цели необходимо решить следующие **задачи**:

- исследовать концепцию электронного документооборота;
- проанализировать структуру современного рынка ПО в области электронного управления документацией, классификацию, сравнение и характеристику данных продуктов;
- обосновать использование вычислительной техники и инструментальных средств разработки;
- охарактеризовать входную, постоянную, промежуточную и результатную информацию;
- формализовать расчеты и построить инфологическую модель;

- разработать структурную часть корпоративной системы электронного управления документацией в рамках рассматриваемой организации.

Разрабатываемая СЭД должна быть реализованной как автоматизированное рабочее место руководителя образовательного учреждения и его секретаря и должна предполагать расширение исполняющих функций.

Информация для исследования была получена по результатам практики, из нормативной, учебно-методической литературы, периодических изданий, Интернет-источников.

Практическая ценность данной работы заключается в разработке автоматизированного рабочего места административного работника учебного заведения, позволяющее фиксировать входные документы согласно принятой классификации, контролировать их движение, осуществлять быстрый поиск, вывод на печать и передачу в архив. Внедрение ПО обеспечит снижение количества тривиальных ошибок типа дублирования информации и освободит сотрудников от рутинной повседневной работы с фиксации выписок и копий документов, значительно сократив время, затрачиваемого на эту работу.

В основе разработки должна быть использована клиент-серверная архитектура хранения данных.

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Основные понятия документооборота

Документы – это основные информационные ресурсы каждой организации, обеспечивающие информационную поддержку принятия управленческих решений на всех уровнях и сопровождающие ее бизнес-процессы.

Документооборот – движение документов в организации от момента их создания или получения и до завершения исполнения или отправки, комплекс работ с документами: прием, регистрация, рассылка, контроль исполнения, формирование дел, хранение, использование документации, справочная работа. Тысячи листов документов, длительный поиск нужного, потери, дубликаты, задержки с отправкой и получением, ошибки персонала составляют далеко не полный перечень проблем, возникающих при неэффективном построении документооборота. Всё это может сильно затормозить, а в исключительных случаях – полностью парализовать работу организации.

Эффективный документооборот является обязательной составляющей эффективного управления. Это непрерывный процесс, объективно отражающий деятельность организации и позволяющий оперативно ей управлять. Документооборот исключительно важен для правильной организации финансового и управленческого учёта.

Автоматизация документооборота позволяет существенно повысить управляемость хранением и движением практически любых типов информации в организации, обеспечить прозрачность и управляемость процессами организации и вывести на качественно новый уровень контроль их выполнения сотрудниками.

Система электронного документооборота позволяет организовать любое взаимодействие между сотрудниками учреждения на основе документов. Она обеспечивает создание документов, их движение по организации, контроль

исполнения документов и процессов, описываемых с их помощью, а также сохранения документов.

Электронный документ (ЭД) – документ, созданный с помощью средств компьютерной обработки информации, который может быть подписан электронной цифровой подписью и сохранён на машинном носителе в виде файла соответствующего формата.

Электронный документооборот (ЭДО) – единый механизм по работе с документами, представленными в электронном виде, с реализацией концепции «безбумажного делопроизводства».

Каждый документ в системе электронного документооборота имеет набор однозначно характеризующих его реквизитов (часто этот набор называют карточкой документа). Карта может включать тип документа, имя автора, даты формирования и изменения документа, отдел, в котором должен храниться документ и тому подобное. Хранение и поиск документов осуществляется также в соответствии с реквизитами, указанными в карточке.

Независимо от размера учреждения, системы документооборота позволяют оперативно изменять формы документов, находящихся в обращении этого заведения. Например, чтобы изменить форму заявки необходимо просто внести соответствующие изменения в ее шаблон, который хранится в системе. Сотрудники, заполняющие такие заявки, не могут подать документ устаревшей формы, поскольку заполнение и передача заявки также осуществляется в системе.

Каждый документ, который создается и обрабатывается в организации, должен пройти несколько установленных стадий (этапов). Например, корпоративный бюджет должен быть составлен, утвержден и выполнен в срок. После этого план и фактические показатели этого бюджета анализируются и направляются в архив. Системы электронного документооборота позволяют жестко контролировать жизненный цикл документов с учетом требований корпоративной среды, а также отраслевых стандартов и законодательства.

С помощью системы электронного документооборота можно организовать коллективную работу над документом. Причем в ней могут принимать участие специалисты, которые находятся в разных отделениях и даже в разных городах.

Системы документооборота позволяют хранить множество версий одного документа. Кроме прямого изменения документов, системы электронного документооборота позволяют аннотировать документы, то есть вставлять примечания и комментарии к ним без изменения самого документа.

Важной особенностью систем электронного документооборота является возможность подписывать документы с помощью электронной подписи и зашифровывать их. Это позволяет не только подтверждать действительность документа и однозначно определять его автора, но и обеспечить безопасность хранения корпоративной информации [1].

Если при отсутствии системы электронного документооборота сотрудник, участвующий в формировании того или иного документа, обязан четко знать, кому этот документ должен быть передан (его маршрут), то при использовании такой системы документ автоматически передается нужному лицу. Маршрутизация документов бывает жесткой и свободной. При свободной маршрутизации сотрудник может направить документ всем адресатам. При жесткой маршрутизации пути документов регламентированы, но в особых случаях действие заранее определенных в системе правил движения документа может приостанавливаться (например, при необходимости отправки проекта документа на рецензию внешнему эксперту).

Чтобы обеспечить целостность информации, находящейся в обороте внутри организации, системы электронного документооборота, как правило, должны и могут быть интегрированы с другими системами управления заведением – бухгалтерскими, учебными, производственными, финансовыми, аналитическими и тому подобное.

Документы из всех систем, которые логически связанные с данным документом, называются контекстом. Контекст может включать содержание электронной и бумажной переписки, аудио- и видеозаписи, факсы, протоколы заседаний, платежные документы и т.д. Контекст может присоединяться к документу вручную или автоматически.

Пользователь, работающий с документами, при необходимости может открыть и просмотреть различные документы непосредственно в системе.

Существует несколько вариантов разграничения полномочий сотрудников компании и контроля доступа к документам. Так, сотрудник может иметь следующие права:

- полный контроль документа;
- создание новых версий документа, но не редактирования;
- аннотирования документ;
- просмотр документа.

Таким образом, сформулируем основные принципы электронного документооборота:

- однократная регистрация документа, позволяющая однозначно идентифицировать документ в любой инсталляции данной системы;
- возможность параллельного выполнения операций, позволяющая сократить время движения документов и повышения оперативности их исполнения;
- непрерывность движения документа, позволяющая идентифицировать ответственного за исполнение документа (задачи) в каждый момент времени жизни документа (процесса);
- единая (или согласованная распределённая) база документной информации, позволяющая исключить возможность дублирования документов;
- эффективно организованная система поиска документа, позволяющая находить документ, обладая минимальной информацией о нём;

- развитая система отчётности по различным статусам и атрибутам документов, позволяющая контролировать движение документов по процессам документооборота и принимать управленческие решения, основываясь на данных из отчётов.

Основные преимущества внедрения электронного документооборота:

- сокращение финансовых и временных издержек по созданию и обработке документов;
- эффективная организация документооборота дает повышение оперативности деятельности подразделений и служб компании;
- интеграция информации, содержащей данные в различных документах, для принятия руководством оперативных и обоснованных управленческих решений;
- создание условий для перехода от рутинного бумажного документооборота к перспективной безбумажной (электронной) технологии;
- оптимизацию бизнес-процессов связанных с делопроизводством и документооборотом на предприятии.

Данный подход к автоматизации предприятий является одновременно и конструктивным и универсальным, обеспечивая автоматизацию документооборота и всех бизнес-процессов предприятия в рамках единой концепции и единого программного инструментария.

1.2 Классификация систем документооборота

Системы управления электронными документами могут быть классифицированы по следующим признакам (рис. 1.1):

- масштаб;
- архитектура;
- преобладающий принцип работы с электронными документами;
- разработчик используемой системы управления базами данных.



Рисунок 1.1 – Классификация систем управления ЭД

По *масштабу* различают корпоративные СУД, СУД для средних предприятий и СУД для организации персонального документооборота. Для корпоративных СУД характерно привлечение фирм-системных интеграторов для адаптации СУД на предприятии, высокие требования к аппаратным и программным средствам, широкий диапазон выполняемых функций, реализация информационных технологий, составляющих основу СУД на «промышленном» уровне, высокая производительность и цена. Поэтому приобретение корпоративной СУД могут позволить себе только крупные предприятия. Корпоративные системы являются мощным инструментом автоматизации делопроизводства и в наибольшей мере способствуют переходу на безбумажный документооборот. Затраты на внедрение корпоративной СУД достигают большой величины [1].

По *архитектуре* различают системы-конструкторы, системы, функционирующие на платформе мощных программ, системы-приложения.

Системы-конструкторы содержат средства, позволяющие

автоматизировать документооборот любого предприятия независимо от профиля его деятельности. Подобные СУД построены на основе принципов модульности, масштабируемости и открытости. Корпоративная система является, как правило, и системой-конструктором.

Системы, работающие на платформе мощных программ, позволяют использовать их функционал в контексте автоматизации документооборота. В качестве платформы могут выступать корпоративная СУД или комплекс интегрированных программ, предназначенный для автоматизации управления предприятием в целом. Особенность систем данного класса состоит в том, что для их работы требуется установить комплекс программ, составляющих соответствующую платформу, что требует дополнительных затрат. То есть стоимость такой СУД складывается из стоимости самой программы и ее «платформы». В данном случае к проблеме выбора системы добавляется проблема выбора программной платформы. Преимущества и недостатки такого решения частично схожи с первым пунктом классификации – с одной стороны, – мощность, с другой, – возможная избыточность и высокая цена.

Системы-приложения представляют собой самостоятельный продукт, не требующий специальных программных средств. Обычно такие продукты относительно просты в установке и освоении.

Классификация СУД по *преобладающему принципу работы с электронными документами* связана с разделением комплекса работ с документами на две части: хранение документов и организация движения документа от одного субъекта к другому. Поэтому СУД могут быть разделены на системы управления документами и системы управления деловыми процессами. Первые – реализуют функции электронного архива и организации совместной работы над документом (groupware), вторые – прохождение документов-заданий между сотрудниками по заранее определенному маршруту. Документы-задания инициируют деловые процессы, связанные с их выполнением, поэтому данные системы называют системами управления

деловыми процессами (workflow).

Системы управления деловыми процессами предназначены для создания сложных прикладных систем коллективной обработки документов в процессе осуществления конкретных бизнес-процессов. Документальные потоки на предприятии привязываются к существующим бизнес-процессам и регламенту их взаимодействия. При жесткой маршрутизации документа заранее прописывается движение документа по всем рабочим местам. Определяются права пользователей на документ в каждой точке маршрута. При свободной маршрутизации исполнитель может определить дальнейший путь движения документа, обычно на одном уровне.

Основу СУД составляют базы данных, в которых хранятся документы, значения их реквизитов, индексированные тексты документов, необходимые для организации контекстного поиска, информация о настройках пользователей, информация, необходимая для администрирования системы.

СУД может содержать собственную систему управления базами данных. То есть ее разработчиком является производитель данной СУД. Может оказаться, что мощности встроенной СУБД недостаточно, чтобы эффективно управлять хранением большого объема данных.

СУД, ориентированные на обработку большого количества документов, используют для хранения информации промышленные СУБД типа MS SQL Server, Oracle, Sybase, Informix.

В соответствии с количеством реализуемых функций СЭД делятся на:

– *Системы с развитыми средствами хранения и поиска информации* (электронные архивы – ЭА). Электронный архив – это частный случай системы документооборота, ориентированный на эффективное хранение и поиск информации. Некоторые системы особенно выделяются за счет развитых средств полнотекстового поиска: нечеткий поиск, смысловой поиск и т.д., другие – за счет эффективной организации хранения: HSM, поддержка широкого диапазона оборудования для хранения информации и т.д.

– *Системы с развитыми средствами workflow (WF).* Эти системы в основном рассчитаны на обеспечение движения неких объектов по заранее заданным маршрутам (так называемая «жесткая маршрутизация»). На каждом этапе объект может меняться, поэтому его называют общим словом «работа» (work). Системы такого типа называют системами workflow – «поток работ» (к сожалению, для этого термина нет точного эквивалента в русском языке). К работам могут быть привязаны документы, но не документы являются базовым объектом этих систем. С помощью таких систем можно организовать определенные работы, для которых заранее известны и могут быть прописаны все этапы.

– *Системы, ориентированные на поддержку управления организацией и накопление знаний.* Эти «гибридные» системы, которые обычно сочетают в себе элементы двух предыдущих. При этом базовым понятием в системе может быть как сам документ, так и задание, которое нужно выполнить. Для управления организацией нужна как «жесткая», так и «свободная» маршрутизация, когда маршрут движения документа назначает руководитель («расписывает» входящий документ), поэтому обе технологии в том или ином виде могут присутствовать в таких системах. Эти системы активно используются в государственных структурах управления, в офисах крупных компаний, которые отличаются развитой иерархией, имеют определенные правила и процедуры движения документов. При этом сотрудники коллективно создают документы, готовят и принимают решения, исполняют или контролируют их исполнение.

При внедрении таких систем на крупных предприятиях важно определить, предоставляет ли система возможность эффективного администрирования, обработки больших объемов информации, интеграции с автоматизированными системами управления производством, масштабируемости, поэтапного внедрения, учета территориальной распределенности, сложной организационной структуры, ролевого принципа организации доступа и т.д.

– *Системы, ориентированные на поддержку совместной работы (collaboration).* Это новое веяние в области систем документооборота, связанное с пониманием изменчивости рыночных условий в современном мире и с необходимостью иметь для быстрого движения «только самое нужное», без лишнего, очень полезного, но тяжелого балласта. Такие системы, в противоположность предыдущим, не включают понятия иерархии в организации, не заботятся о какой-либо формализации потока работ. Их задача – обеспечить совместную работу людей в организации, даже если они разделены территориально, и сохранить результаты этой работы. Обычно реализованы в концепции «порталов». Они предоставляют сервисы хранения и публикации документов в intranet, поиска информации, обсуждения, средства назначения встреч (как реальных, так и виртуальных). Такие системы находят заказчиков среди быстро развивающихся коммерческих компаний, рабочих групп в крупных фирмах и государственных структурах.

Системы, имеющие развитые дополнительные сервисы. Например, сервис управления связями с клиентами (CRM – Customer Relation Management), управления проектами, биллинга, электронной почты и пр. (Отметим, что по сложности функций система документооборота и, например, сервис CRM могут иметь различные пропорции в зависимости от организации. Но в контексте этой статьи функциональность CRM является дополнительной).
Типовые требования к СЭД

Если следовать букве стандарта на составление технического задания, требования, которые типовой пользователь может предъявить к типовой системе электронного документооборота, можно описать следующим образом.

Система электронного документооборота должна:

- обеспечивать надежное хранение документов и их описаний;
- обеспечивать жизненный цикл документа (его создание, хранение версий, публикация, блокировка доступа к изъятому документу, передача документа для хранения в архиве);

- допускать задание пользователем различных типов документов, создания и редактирования карточек для них;
- поддерживать иерархию категорий для эффективного поиска документа;
- осуществлять поиск документов на основе информации из карточки, а также полного текста;
- обеспечивать разделение доступа к документам на уровне отдельных пользователей, по ролевому принципу, и на основе иерархической структуры организации;
- поддерживать технологию HSM;
- протоколировать все события, связанные с работой пользователей и самой системы; необходимо наличие развитых средств администрирования;
- поддерживать удаленный доступ к информации.

Продвинутые системы должны поддерживать:

- кластерные технологии для обеспечения бесперебойной работы;
- территориально распределенные организации;
- алгоритмы шифрования при хранении и передаче данных;
- цифровую подпись.

Требования к архитектуре:

- наличие выделенного сервера приложений;
- наличие тонкого клиента; поддержка доступа к документам с использованием браузера.
- многоплатформенность для обеспечения масштабируемости;

Требования к открытости и интеграции с другими системами:

- интеграция со средствами потокового ввода документов;
- интеграция с офисными приложениями;
- интеграция с электронной почтой;
- наличие развитого программного интерфейса (API);

- интеграция со стандартными службами каталогов (к примеру, LDAP) для ведения и синхронизации списка пользователей системы;
- возможность адаптации пользовательского интерфейса под конкретные задачи;
- возможность дополнения системы собственными специализированными компонентами;

В случае использования внешней базы данных для хранения атрибутов документов необходимо наличие подробного описания структуры данных и средств работы с разными СУБД.

1.3 Автоматизация документооборота в образовательном учреждении

В современных условиях автоматизация административного управления в образовательном учреждении – объективная необходимость и является средством, с помощью которого обеспечивается выполнение функций управления: планирование, составление расписаний и отчетности, оперативное и диспетчерское управление и т.п. Эти функции объективно необходимы для обеспечения качественного учебного процесса и нормальных условий работы персонала.

Автоматизация административного управления направлена на более эффективное и качественное выполнение управляющих функций. Включение средств автоматизации в процесс управления заведением не изменяет целей системы управления, ее функций, а меняет лишь способы достижения этих целей и выполнения установленных функций. Таким образом, работы по автоматизации управления направлены на включение машин в поток работы, выполняемой системой управления, с целью повышения эффективности ее деятельности.

Разработка систем управления с использованием средств автоматизации имеет ряд особенностей. Одна из них связана с тем, что в подавляющем большинстве случаев автоматизации подлежат уже работающие системы

управления и автоматизация не должна нарушать выполнение их функций. Это общее требование относится ко всем способам использования ПК. Структуры, методы работы, формы документов этих систем рассчитаны на «ручное» выполнение, то есть на возможности людей. Насколько глубоко изменения, связанные с введением ПК, коснутся существующей системы, зависит от ряда факторов – возможностей разработчиков, наличия вычислительных устройств, поставленных целей, определенных сроков, выбранного способа использования вычислительных машин.

Поскольку внедрение ПК направлено на повышение эффективности управления, поэтому независимо от выбранного способа нужно определить источник повышения эффективности. Для этого нужно определить операции, которые целесообразно выполнять с помощью ПК, что в свою очередь требует достаточно детального изучения работы существующей неавтоматизированной системы управления: определение состава подразделений системы управления (организации), связи между ними, разделение ответственности между подразделениями и выполняемую ими работу[3] .

Машинное выполнения работ требует прежде всего разработки алгоритмов и соответствующего программного обеспечения. Количество необходимых алгоритмов и программ, их сложность зависят от выбранного способа использования ПК.

Но в любом случае алгоритмическое и программное обеспечение разрабатываются постепенно, в установленной очередности для каждой подсистемы.

Перевод «ручной» системы в автоматизированную происходит постепенно, поскольку даже при небольших по масштабам работах невозможно охватить одновременно все задачи управления и автоматизировать их выполнение. Количество и объем частей работ целиком определяются имеющимися ресурсами разработчиков. При определении очередности работ

используют возможность разделения системы управления на части – подсистемы.

Существуют два вида подсистем – структурные и функциональные. Структурные соответствуют организационной структуре системы управления и используют в основном для организации разработок.

Структурно АИС образовательного учреждения должна состоять из системы автоматизированных рабочих мест (АРМ) функционально увязанных между собой с помощью локальной вычислительной сети учреждения (ЛВС). ЛВС обеспечивает через ее сервер взаимодействие рабочих автоматизированных мест, имеющих разные права доступа к базам данных (рис.1.2).

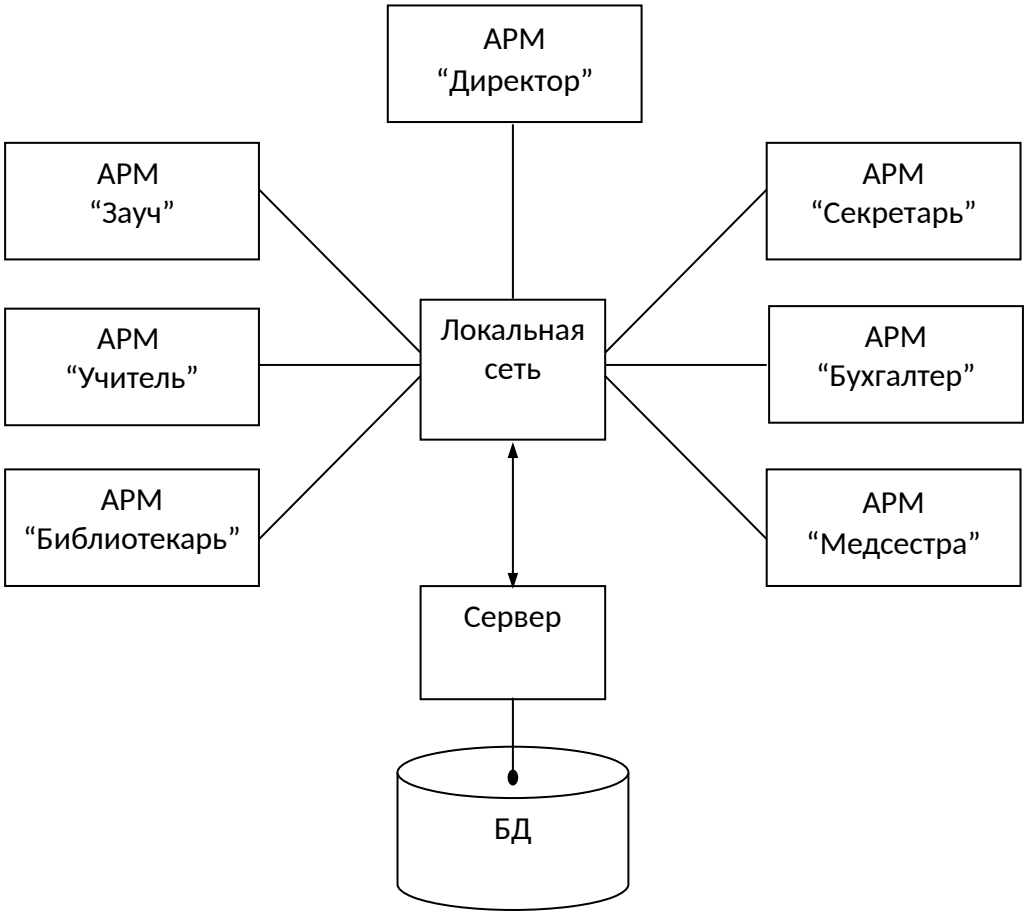


Рисунок 1.2 – Модель перспективной организации информационного пространства образовательного заведения с централизованной БД.

При наличии в учреждении уже отдельных АРМ, их интеграция в общую информационную систему заведения может осуществляться путем разработки распределенной системы управления БД.

Понятно, что решение проблемы взаимодействия БД в узлах усложняется, если узлы распределенной системы управления БД основанных на различных программно-технических платформах, поддерживают различные модели данных.

Функциональные подсистемы соответствуют основным функциям управления, выполняемым системой (планирование, оперативное управление, управление кадрами и т, д.).

Подсистемы автоматизированной информационно-аналитической системы управления образовательным учреждением изображены на рис. 1.3.

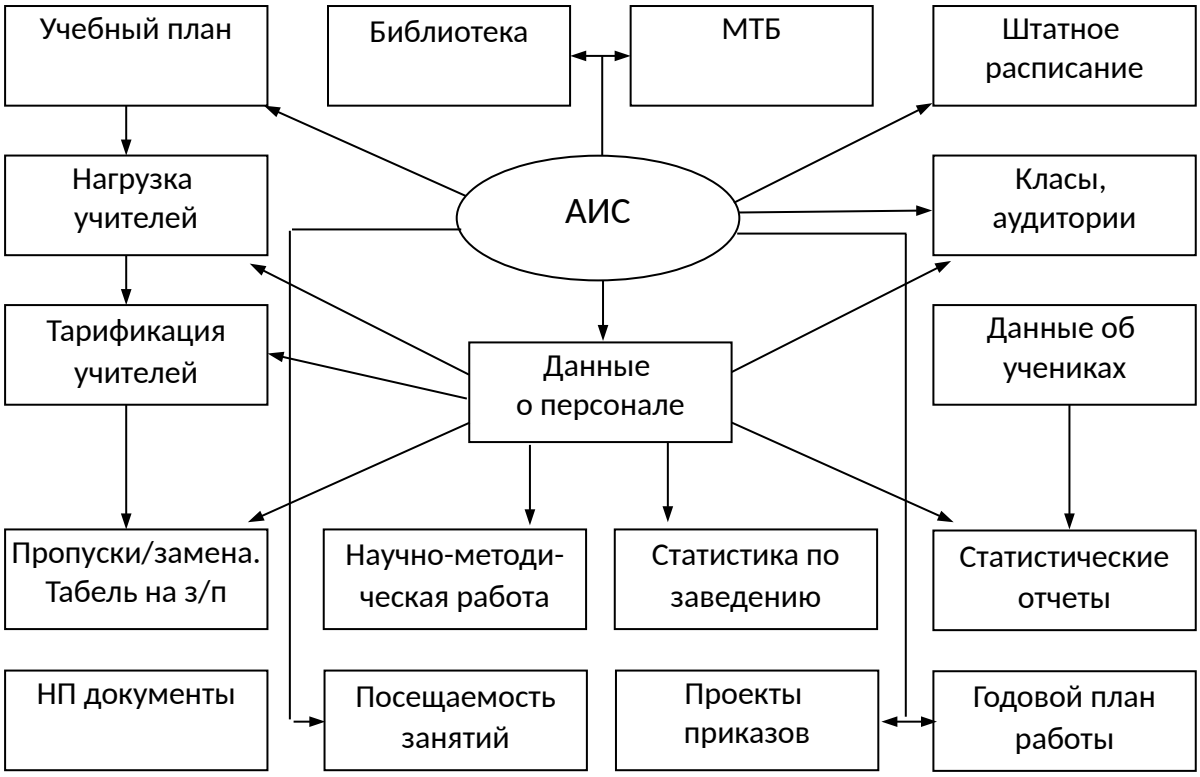


Рисунок 1.3 – Функциональные подсистемы АИС управления образовательным учреждением

Автоматизация управления осуществляется именно путем автоматизации выполнения отдельных функций, причем эта автоматизация также осуществляется постепенно (рис. 1.4).

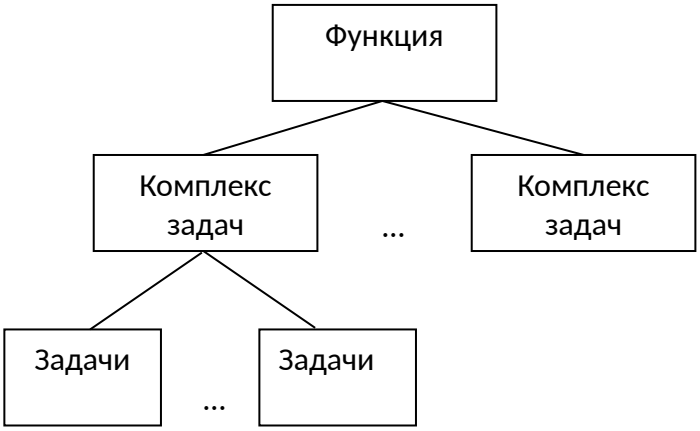


Рисунок 1.4 – Схема функции управления

Для выполнения любой функции необходимо выполнить **n** комплексов работ (n>1). Каждый комплекс в свою очередь состоит из t задач. Автоматизация выполнения функций и осуществляется переводом на машинное выполнение отдельных функциональных задач. Для них разрабатываются модели, программы, создаются информационные массивы. Введение автоматизации отдельных функциональных задачах должно осуществляться таким образом, чтобы обеспечить стыковку отдельных задач в комплексы, а комплексы – в автоматизированную функцию и т.д.

Таким образом, при автоматизации административного управления, с одной стороны, имеет место стремление создать единую законченную систему, а с другой – ограниченные возможности заставляют вести внедрение ЭВМ постепенно, наращивая количество работ, выполняемых с помощью машин. Гибкость систем административного управления, то есть возможность их адаптации к изменениям является непременным условием успешного управления.

Внедрение системы автоматизации управления документами всегда предполагает создание корпоративного хранилища документов. Логическая структура этого хранилища, иерархия хранения документов, разработка и реализация политики безопасности в рамках хранилища требуют особой тщательности и выполняются на самом раннем этапе проекта внедрения.

При этом необходимо учесть, что функция сохранения документов может быть развернута в самом начале проекта. Поэтому уже на первом этапе внедрения можно переместить все документы из существующих мест хранения (как правило, локальных или дисков сети) в единое корпоративное хранилище. Это даст возможность пользователям ознакомиться с базовыми функциями системы и постепенно освоиться с ее использованием. Повысить скорость адаптации сотрудников к внедряемой системы позволяет также использование сервисных функций, таких как оповещение об изменении документа, о появлении нового документа и т.п.

Система электронного документооборота позволяет создать единое информационное поле, в котором могут работать все сотрудники заведения. Поэтому необходимо уделить особое внимание описанию процедур доступа к информации. При этом следует найти разумный компромисс между требованиями безопасности и полнотой информации, который должен обладать каждый отдельно взятый сотрудник.

Реализация информационной модели управления общим образованием осуществляется на базе серверных баз данных: Microsoft SQL Server, Oracle, MySQL, InterBase, FireBird. При разработке базы данных предполагается создание и введение справочников подразделений, преподавательского состава, ученических (студенческих) групп, учебных дисциплин и курсов, учебных аудиторий, распределения учебной нагрузки.

1.4 Общая характеристика объекта исследования

Исследование о возможности разработки и внедрения автоматизированной системы электронного документооборота в образовательном заведения проведено на базе государственного бюджетного общеобразовательного учреждения города Москвы "Школа № 2095 "Покровский квартал".

Основной деятельностью учреждения является выполнение работ, оказание услуг в целях обеспечения реализации предусмотренных нормативно-правовыми актами РФ и города Москвы полномочий в сфере образования. Осуществляется образовательная деятельность по общеобразовательных программам начального общего, основного общего, среднего общего образования.

Учреждение создано 14 января 2014 г в порядке реорганизации в форме слияния:

государственного бюджетного образовательного учреждения города Москвы средней общеобразовательной школы с углубленным изучением французского языка №1225,

государственного бюджетного образовательного учреждения города Москвы средней общеобразовательной школы № 613,

государственного бюджетного образовательного учреждения города Москвы средней общеобразовательной школы № 330,

государственного бюджетного образовательного учреждения города Москвы детского сада комбинированного вида № 2334,

государственного бюджетного образовательного учреждения города Москвы детского сада комбинированного вида № 1983,

государственного бюджетного образовательного учреждения города Москвы центра развития ребенка-детского сада № 1948,

государственного бюджетного образовательного учреждения города Москвы детского сада общеразвивающего вида № 275.

Учредителем образовательного учреждения является Департамент образования города Москвы.

Местонахождение организации: 105064, Москва, Большой Казённый переулок д.9. Филиалов учреждения нет.

Численность сотрудников организации: специалистов (в том числе совместителей) 394 чел., постоянные (основные) сотрудники 366 чел.

1.5 Обзор существующих аналогов и разработок

В России в настоящее время наступил качественный перелом: получило общественное признание положение о том, что управление электронным документооборотом – одна из основных частей информационной системы любого предприятия. По данным CNews Analytics, рынок СЭД в последние годы является одним из самых динамично развивающихся сегментов отечественной ИТ-индустрии. Основным потребителем СЭД является государственный сектор, и именно интерес со стороны государства стал основой устойчивости рынка СЭД [11].

Но, несмотря на это, Россия в области управления электронными документами значительно отстает от зарубежных стран. Дело в том, что деятельность государственных органов в данной области активизировалась только в последние три-четыре года, отя история российского электронного документооборота начинается еще в 1980-х гг.[13] Нормотворческая деятельность в этой области началась намного позже.

Приведем сравнительные характеристики наиболее широко используемых сегодня в России систем электронного документооборота. Стоимость лицензий на ПО и работ по внедрению и техническому сопровождению, в каждом конкретном случае может изменяться в зависимости от "набора модулей и функционала" и поэтому для

уточнения стоимости интерсуемого решения следует обращаться к разработчикам или их уполномоченным представителям.

Продукты упорядочены по возрастанию общей стоимости.

Таблица 1.1 Стоимости внедрения СЭД

Наименование СЭД	Стоимость 5 лицензий + полный функционал, $	Стоимость внедрения, $/час	Стоимость обучения, $/час
МОТИВ	230,00	33,00	21,00
Effect Office	630,00	17,86	18,00
ДЕЛО	1 025,00	37,06	26,00
jDocflow	1 000,00	0,00	0,00
ИНТАЛЕВ-Документооборот	1 750,00	0,00	0,00
ЕВФРАТ-Документооборот	2 740,00	0,00	15,00
DIRECTUM	3 885,00	20,00	12,00

Как видно из характеристик этих программных продуктов данные системы являются рентабельными лишь для крупных комплексов, так как цена для одного рабочего места очень высока.

При исследовании функциональности продуктов были выбраны наиболее объективные критерии функциональности: платформа для хранения информации и элементы защиты данных.

Предпочтительней, когда программа имеет клиент-серверную архитектуру, в качестве СУБД используется SQL-сервер, входящий в комплект поставки. SQL-серверная архитектура дает несравнимую масштабируемость,

транзакционная модель доступа к данным гарантированную надежность всей системы в ответственные моменты, а чрезвычайно широкие возможности разграничения прав доступа позволят защитить данные.

Таблица 1.2 Функциональные характеристики СЭД

Наименование СЭД	Тип "электронного документа" в СЭД	Хранилище документов	Используемые ЭЦП
МОТИВ	Файл любого расширения	FreeBSD и/или файловое хранилище в сети	(+) CryptoPRO
Effect Office	Файл + текстовые поля на карточке + дополнительные атрибуты	MS SQL server	(-) можно встроить
ДЕЛО	Файл любого расширения	БД системы ДЕЛО либо файловые каталоги	(+) CryptoPRO
jDocflow	Файл любого расширения	СУБД MS SQL Server и/или файловые каталоги	(-) можно встроить
ИНТАЛЕВ-Документооборот	Файл любого расширения	СУБД MS SQL Server и/или файловые каталоги	(-) можно встроить
ЕВФРАТ-Документооборот	Файл + текстовые поля на карточке + дополнительные атрибуы	СУБД "Ника".	(+) нет данных
DIRECTUM	Файл и поля на	СУБД	(+)

	учетной карточке	MS SQL Server	CryptoPRO, LanCrypto

Среди недостатков рассмотренного программного обеспечения отметим сложность настройки и конфигурирования. Также стоимость остается достаточно высокой в условиях сегодняшней экономической нестабильности.

Многие системы управления документами имеют сложный, иногда неинтуитивный интерфейс пользователя. В нашем же случае необходимо приложение, которое позволяет вести полноценный учет документации, оптимизировать труд администратора и при этом совершенно не требует от пользователя глубоких познаний в области управления системами электронного документооборота.

1.6 Постановка задачи

Основными задачами документооборота в исследуемом учреждении являются учет нормативно-правовых актов, распорядительных документов вышестоящих органов, издаваемых внутри учреждения приказов и выдачи документов на их основании, организация архива документов. Разрабатываемая программа должна ускорить и оптимизировать работу секретаря.

В проектируемой системе решаемые задачи следует реализовать соответствующими программными модулями, каждый из которых будет иметь возможность выполнения ряда операций, таких как корректировка информации, поиск данных, формирование отчетов. Процессы заполнения, корректировки и поиска данных необходимо реализовать однотипным способом во всех модулях. При добавлении и редактировании данных применять контроль правильности ввода пользователем информации, путем ее сравнения с необходимым типом, диапазоном изменения возможных значений, маской ввода и тому подобное.

Для добавления и редактирования данных использовать экранные формы со всеми необходимыми полями ввода, а также пояснениями и управляющими элементами (например, кнопками), предназначенными для осуществления управляющих воздействий (сохранение, отмена изменений) и навигации (перемещения) по БД.

Для обеспечения поиска данных также использовать экранные формы, позволяющие задавать различные диапазоны значений необходимой информации, с контролем допустимости значений условий поиска. Поиск информации следует предусмотреть как по отдельным полям таблиц, так и по совокупности полей, на частичный или полный совпадение условий поиска, без учета регистра букв. Для удобства пользователя реализовать возможность указания диапазонов изменения значений для многочисленных полей и полей типа «дата». В каждом из программных модулей системы предусмотреть наличие всех необходимых отчетных форм для формирования и печати документов установленной формы. Все отчеты генерировать автоматически, используя выборки информации из БД.

Источником оперативной информации для реализации решения поставленной задачи должны быть входные документы секретаря учреждения. Эти данные постоянно будут меняются, поэтому отличаться большим объемом обрабатываемой информации. Постоянная же информация должна храниться в виде таблиц и реализовываться с помощью специальных справочников.

2. ПРОЕКТНАЯ ЧАСТЬ

2.1 Обоснование выбора обеспечения реализации проекта

2.1.1 Информационное обеспечение

Рассмотрим основные типы документов, используемых в работе учреждения и их характеристики. Типов документов достаточно много (на отдельных предприятиях их число доходит до 500-600). В то же время, с точки зрения канцелярии основных видов документов бывает очень небольшое количество, а именно всего 3 вида:

- Входящие. Это документы, которые поступили в учреждение от внешних источников. Большинство входящих документов должны порождать соответствующие выходные, причем в заранее установленный срок. Сроки устанавливаются нормативными актами, указывающие время ответа на соответствующий входящий документ, или сроком исполнения, указанным непосредственно во входящем документе.

– Исходящие. Большинство исходящих документов являются ответом организации на соответствующие входящие документы. Некоторая часть исходящих документов может готовиться на основе внутренних документов учреждения. Небольшое число исходящих документов может требовать поступления входящих документов (например запросы в сторонние организации типа: "Прошу дать справку по вопросу ... в срок до ...").

– Внутренние. Данные документы используются для организации работы предприятия. Через канцелярию проходят не все внутренние документы, а только переписка крупнейших структурных подразделений учреждения (особенно если они территориально разнесены) и приказы его руководства. Так же через канцелярию проходят внутренние документы, порождающие выходные. В частности, по общим правилам делопроизводства единственный способ отправить запрос, письмо или материалы во внешнюю

организацию это направить внутренний документ в канцелярию, где его превратят в выходной и отправят в стороннюю организацию.

Прежде всего все документы, проходящие через общий отдел обладают уникальным регистрационным номером (могут быть свои алгоритмы построения номеров для каждого из типов документов). Бумага не имеющий регистрационного номера – это просто бумага, а не документ. Документом его делает именно наличие регистрационного номера.

Детализируем атрибуты документов:

1. Регистрационный номер документа однозначно позволяет сослаться на документ, прошедший через канцелярию. Структура регистрационного номера может быть различна для исходящих, входящих и внутренних документов. Структура регистрационного номера определяется в каждой организации самостоятельно.

2. Источник документа (контрагент) указывает откуда получен документ. Для входящих документов это организация, для внутренних и исходящих документов это подразделение или конкретное лицо из руководства предприятия.

3. Ответственный исполнитель документа – сотрудник учреждения, которому поручено выполнение данного документа. Исполнитель документа всегда один и только один. Иногда встречаются два исполнителя документа (для документов длительного выполнения при изменении кадрового состава предприятия).

4. Код документа по номенклатуре дел предприятия относит документ к тому или иному кругу типичных вопросов, решаемых учреждением в своей деятельности. Номенклатура дел – это формальный список дел, который утверждается заранее на определенный период времени (обычно на год). Ранее данная номенклатура была достаточно статична, в современных условиях номенклатура дел является высокодинамичным документом, который может обновляться несколько раз в год.

Далее в документах могут быть атрибуты, специфичные для данного типа документа.

Входящие документы имеют кроме базовых атрибутов еще:

– контрольный срок исполнения. Данный атрибут берется непосредственно из входного документа (где он может быть указан);

– Контролирующий лицо. Обычно это директор или завуч школы, назначающий конкретного исполнителя документа, он же ставит резолюцию "О выполнении доложить".

Внутренние документы помимо стандартных атрибутов могут иметь еще и следующие:

- список подразделений учреждения для ознакомления.
- контрольный срок ознакомления или исполнения.
- список исполнителей документа.

Выходные документы имеют следующие дополнительные атрибуты:

- документ-основание. Этот атрибут указывается всегда, поскольку что исходный документ всегда порождается на основе входящего или внутреннего.
- список рассылки.

Канцелярия отвечает за сроки выполнения входящих документов, точнее выполняет фискальные функции: кто и когда задержал исполнение документа. При этом существует два основных маршрута прохождения документа:

1. непосредственно исполнителю;

2. на принятия решения в администрацию.

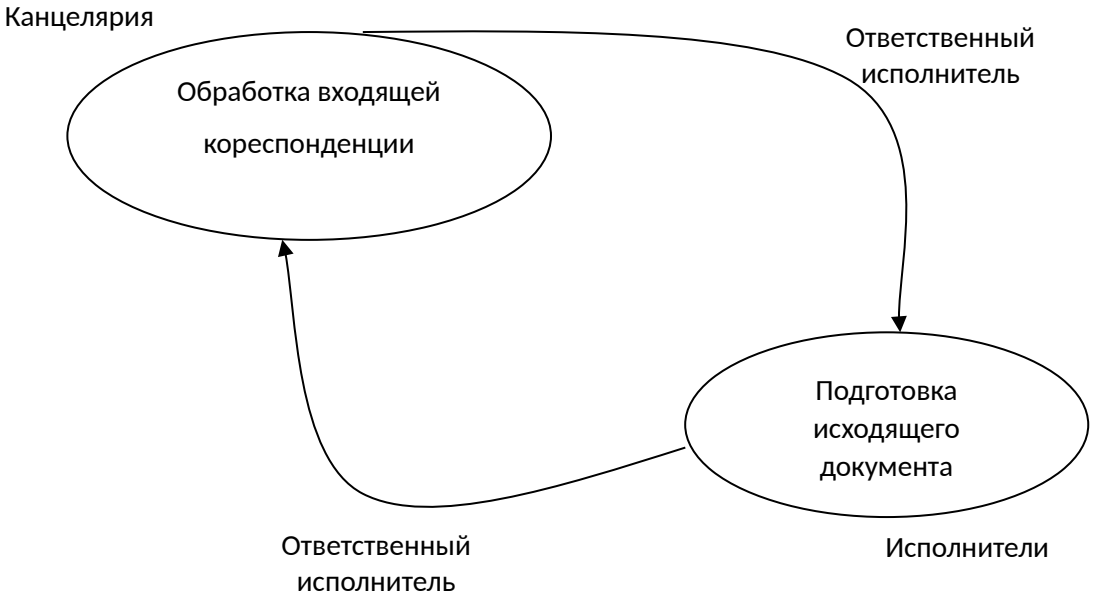


Рисунок 2.1 – Схема прямой обработки входящего документа

Второй маршрут означает, что почти наверняка этот документ будет поставлен на контроль. Для документов, проходящих по второму маршруту, ответственным исполнителем является кто-то из руководителей среднего звена, а директорат осуществляет контроль по результатам работы.

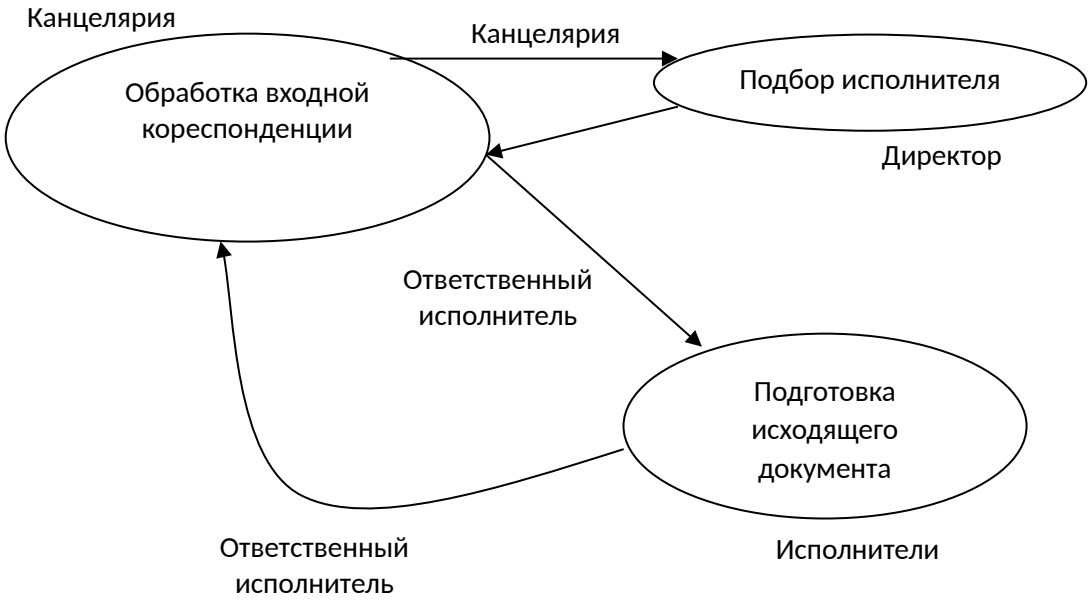


Рисунок 2.2 – Схема обработки с контролем директора

2.1.2 Выбор средств разработки

Выбор программных средств для реализации БД и информационной системы в целом зависит от предметной области и решаемых задач пользователя. Одним из основных критериев выбора СУБД является оценка того, насколько эффективно внутренняя модель данных, поддерживаемая системой, способна описать концептуальную схему. Также как преимущество будем рассматривать сравнительно низкую стоимость, или открытость СУБД.

На текущий момент наиболее распространенными промышленными серверами являются следующие: InterBase и Firebird, Oracle, PostgreSQL, MS SQL 2005, MySQL [8].

Для всех подобных систем используется вполне среднее оборудование для сервера – процессор на уровне 3 ГГц, от 1 до 4 ГБ памяти. Разумеется, производительность конкретной системы зависит от качества приложений. Для информационной модели разрабатываемой системы предполагается использовать СУБД FireBird 2.5.

СУБД FireBird является одной из самых популярных в мире бесплатных, кроссплатформенных систем управления базами данных с открытым исходным кодом. Она была разработана на основе исходного кода СУБД Interbase и развивается сегодня независимым международным сообществом. По надёжности, производительности и функциональным возможностям эта система мало в чём уступает признанным лидерам своего класса – Oracle и Microsoft SQL Server.

Firebird полностью поддерживает стандартны ANSI в синтаксисе языка SQL и может работать под управлением многих операционных систем – Windows, Linux, MacOS, Solaris и различных Unix-платформах. Среди достоинств этой системы использование очень развитого языка для хранимых процедур и триггеров. Также следует отметить, что Firebird полностью бесплатный сервер – не требует ни регистрации, ни оплаты за поддержку.

В качестве преимуществ Firebird можно отметить многоверсионную архитектуру, обеспечивающую параллельную обработку оперативных и аналитических запросов (это возможно потому, что читающие пользователи не блокируют пишущих), компактность (дистрибутив 5Mb), высокую эффективность и мощную языковую поддержку для хранимых процедур и триггеров.

Для разработки информационной системы предложено использовать визуальную среду Borland Delphi 6. Выбор обусловлен тем, что данная версия продукта обладает функционалом, обеспечивающим решения поставленной задачи является стабильной версией в линейке ИСР разработчика. Последние версии продукта (семейство XE) ориентированы в первую очередь на разработку ПО для мобильных устройств, при этом эти ИСР имеют значительную стоимость, что в нашем случае имеет решающее значение.

Выбор языка программирования основывается на следующем: он даёт полномасштабное средство разработки для почти всех известных СУБД, совместимое со стандартами ODBC, IDAPI и к тому же генерируют компактные и легко переносимые исполняемые файлы [4].

Среда визуального программирования Delphi совместима со всеми прикладными алгоритмами, написанными до сих пор на Pascal. Не будет лишним знание интерфейса Windows.

Основные характеристики выбранного языка программирования:

- высокопроизводительный компилятор в машинный код;
- объектно-ориентированная модель компонент;
- визуальное (а значит и быстрое) построение приложений из программных прототипов;
- масштабируемые средства для построения баз данных.

Интегрированная среда разработки IDE Delphi предоставляет всевозможные средства для создания, тестирования и редактирования проекта.

Среда Delphi является одной из наиболее популярных систем разработки приложений. Программа проста в использовании, не требует специальных навыков, кроме знаний основ работы в системе Microsoft Windows. Программа широко использует стандартные элементы управления, что обеспечивает единство интерфейса системы и программного средства, а, следовательно, и удобство использования.

Преимуществами Delphi 6 является то, что для работы с различными данными создается наиболее подходящий для каждого конкретного случая графический интерфейс, содержащий всевозможные меню, множество управляемых окон и разнотипных элементов управления.

2.1.3 Технология доступа к данным

В Delphi есть ряд компонентов, которые предназначены только для работы с Interbase (Firebird), эти компоненты находятся во вкладке INTERBASE. Чтобы обрабатывать некоторую структуру данных для нее должна быть написана программа, поставщик этих данных в соответствии с системными требованиями, такая программа называется Provider. Такие поставщики сегодня реализованы для разных структур данных и различных СУБД. С помощью технологии OLE DB можно однотипным образом обрабатывать сложную и специфическую информацию. Однако работа с OLE DB достаточно сложна, поэтому фирма Borland разработала новую технологию IBExpress (IBX), представляющая собой набор простых компонентов.

Если планируется создать новое приложение, ориентированное на работу с данными и независимой ни от конкретной СУБД и не от способа хранения информации, то лучше использовать соответствующие технологии доступа.

Технология IBX обеспечивает универсальный доступ к источникам данных из приложений БД Firebird. Такую возможность предоставляют функции набора интерфейсов, созданные на основе общей модели объектов COM и описанные в спецификации OLE DB.

Обработка запросов SQL – это основная обязанность сервера. IBX представляет собой набор специализированных объектов COM, стандартных функций обработки данных и специализированных функции конкретных источников данных и интерфейсов, обеспечивающих передачу данных между объектами.

Согласно терминологии IBX, любой источник данных (база данных, электронная таблица, файл) называется хранилищем данных, с которым с помощью провайдера данных взаимодействует приложение.

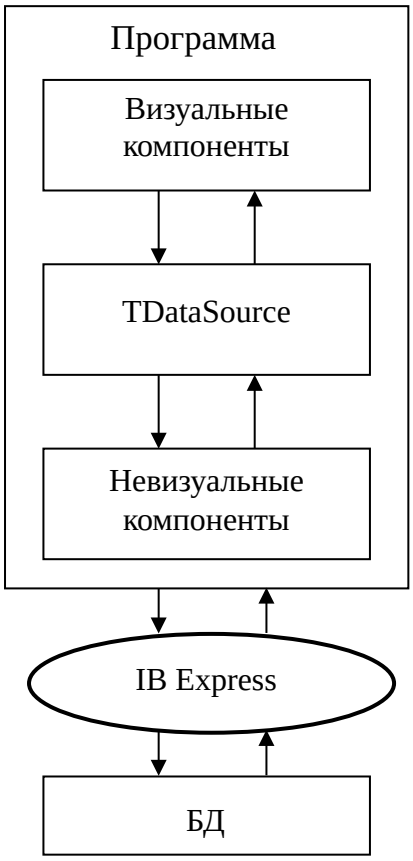


Рисунок 2.3 – Технология доступа к данным [10]

Минимальный набор компонентов приложения может включать объект соединения, объект набора данных, объект процессора запросов.

Механизм доступа к данным через IBX: многочисленные объекты, и интерфейсы реализованы в VCL Delphi в виде набора компонентов, расположенных на странице INTERBASE.

Поскольку технология IBX основана на стандартных интерфейсах COM, которые являются системным механизмом Windows, это сокращает общий объем работающего программного кода и позволяет распространять приложения управления БД без вспомогательных программ и библиотек.

Компоненты доступа к данным является невизуальными компонентами. Таблицы БД располагаются на диске и являются физическими объектами. Набор данных является логической таблицей, из которой можно работать при выполнении приложения. Взаимодействие таблицы и набора данных напоминает взаимодействие физического файла и файловой переменной.

В целом технологию IBX можно охарактеризовать как наиболее современную технологию разработки приложении для работы с распределенными БД по технологии клиент-сервер.

Передача и прием сообщений в виде рассматриваемых пакетов осуществлялась с помощью компонентов – сокетов: TClientSocket и TServerSocket.

Сокет – интерфейс прикладного программирования для сетевых приложений TCP/IP.

По методу работы сокеты делятся на три основных типа:

– Клиентские сокеты. Соединения устанавливаются между клиентским сокетом и серверным сокетом на удаленной машине. Соединения этого типа инициализируются со стороны клиента. Для того, чтобы открыть соединение, клиентский сокет должен “знать” имя или IP-адрес удаленной машины и номер порта, используемый серверным сокетом. Итак, клиент посылает серверу запрос на соединение. Сервер ставит эти запросы в очередь и обслуживает их по мере возможности.

– Слушающие сокеты. Серверные сокеты сами не занимаются установлением связи с клиентскими сокетами. Эту задачу выполняют слушающие сокеты, встроенные в серверный компонент TServerSocket. Когда поступает запрос от нового клиента на подключение, его получает слушающий

сокет и ставит в очередь. После того, как серверный сокет освободится от текущей работы, он обрабатывает запрос из очереди и создает новый сокет для нового соединения. Таким образом, слушающий сокет всегда готов к получению нового запроса на соединение.

– Серверные сокеты. Серверный сокет устанавливает соединение с клиентским сокетом в ответ на его запрос, полученный слушающим сокетом. При этом клиентский сокет получает описание серверного сокета, после чего соединение считается установленным.

При установке связи между двумя приложениями, эти приложения, сокеты являются конечными точками сетевого соединения. Перед открытием соединения двух сокетов необходимо описать обе конечные точки. Для слушающего сокета достаточно указать номер порта, который он слушает. Клиентскому сокету нужно указать имя сервера или его IP-адрес и номер порта серверного сокета. Задавать компьютер по имени проще, чем использовать IP-адрес.

События, возникающие при работе с сокетами, можно условно разделить на четыре группы: события закрытия и открытия соединения, события чтения и записи, и события сообщающие об ошибке.

Обмен данными – это очевидно основная задача установленного соединения сокетов. Приложениями используются неблокирующий тип сокетов (у клиентского сокета свойство ClientType имеет значение ctNoneBlocking, а у серверного сокета свойству ServerType присвоено значение stNoneBlocking), то есть операции чтения/записи происходят асинхронно и не блокируют выполнение программного кода [2].

Поскольку все теоретические аспекты создания требуемой информационной системы рассмотрены, перейдем к реализации проекта.

2.2 Проектирование БД

В базе данных отображается информация об определенной предметной области (ПО). Инфологическая модель (ИМ) – это описание предметной области, выполненное без ориентации на используемые программные и технические средства.

Для решения задач рассмотренных в данной работе разработана инфологическая модель, которая представлена на рис. 2.4.

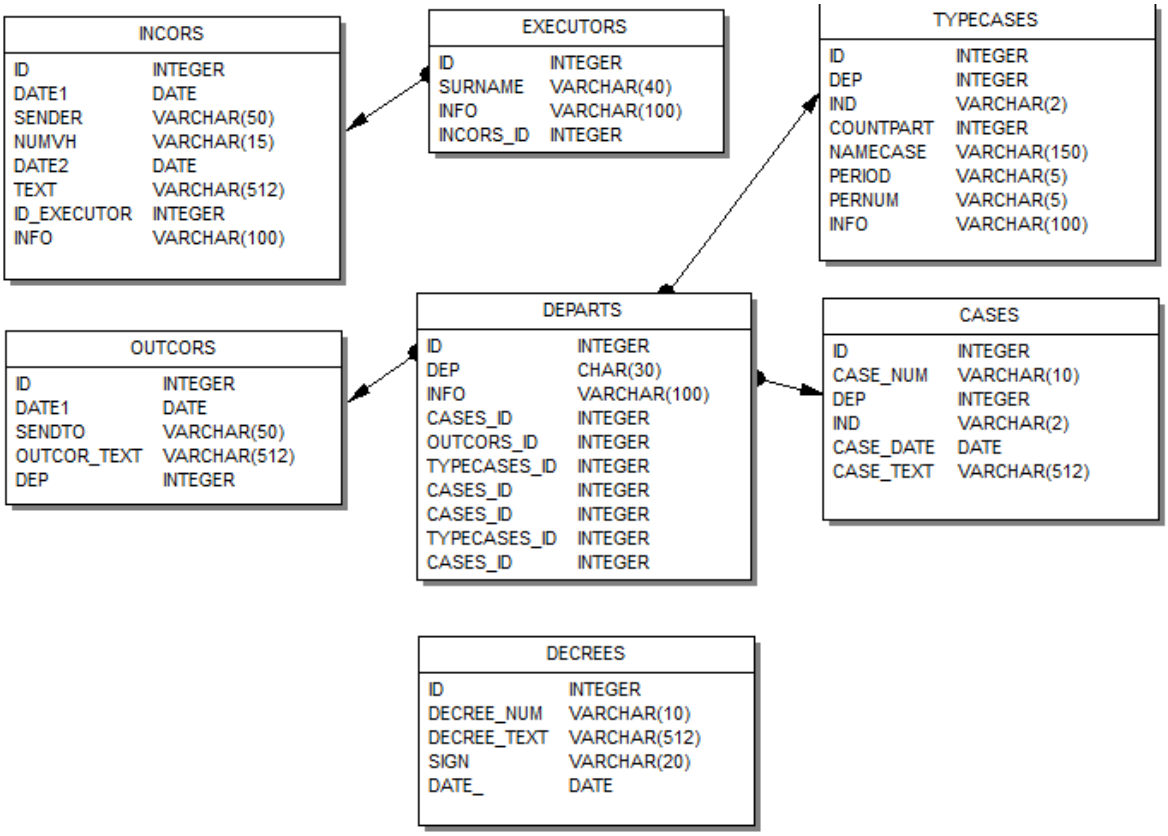


Рисунок 2.4 – Инфологическая модель БД

На основе инфологической модели была спроектирована физическая база данных на платформе СУБД Interbase 6.

Доступ к БД должен производиться из програми с помощью SQL-запросов (3 диалект). Размер страницы обмена составляет 4 096 Байт.

Определим имена основных таблиц, соответствующих объектам инфологической модели.

БД имеет 6 таблиц, которые представлены далее в табл. 2.1–2.8

Таблица 2.1 – Структура БД

Объект	Имя таблицы в БД
Справочник “Подразделения”	Departs
НД “Входящая кореспонденция”	InCors
НД “Исходящая кореспонденция”	OutCors
НД “Приказы”	Decrees
НД “Дела”	Cases
Справочник “Типы дел”	TypeCases

Определим структуру таблиц БД.

Для описания полей таблицы в БД определены следующие домены:

Таблица 2.2 – Определения доменов БД

Домен	Тип	Размер	Ограничения
TCount	Integer	4	Значение по умолчанию – 0
TDate	Char(10)	10	Маска ввода – '_ _ ._ _._ _ _ _'
TDep	Char(30)	30	
TDepNum	Integer	2	
TExecuter	Varchar(20)	20	
TIndexCase	Varchar(2)	2	Обязательное
TInfo	Varchar(100)	100	

Таблица 2.2 – *Продолжение*

Домен	Тип	Размер	Ограничения
TDecreeNum	Varchar(10)	10	Обязательное
TDecreeText	Varchar(512)	512	
TNameCase	Varchar(150)	150	
TNumber	Integer	4	Обязательное
TNumVH	Varchar(15)	15	
TPerNum	Varchar(5)	5	
TSave	Varchar(5)	5	
TSender	Varchar(50)	50	
TSign	Varchar(20)	20	
TCaseNum	Varchar(10)	10	Обязательное
TCaseText	Varchar(512)	512	

В таблицах символом # отмечают ключи (первичные индексы), а символом (#) – вторичные индексы [8].

Далее представлено описание входящей оперативной информации и ее структуры.

При получении первичных документов заполняются следующие таблицы:

Таблица 2.3 – Структура таблицы «Приказы» (DECREES)

Поле	Ключ	Тип	Описание поля
ID	#	TNumber	Код
Decree_Num		TDecreeNum	Номер приказа
Decree_Text		TDecreeText	Текст приказа
Sign		TSign	Контроль
Data		TDate	Дата приказа

Таблица 2.4 – Структура таблицы «Подразделения» (DEPARTS)

Поле	Ключ	Тип	Описание поля
ID	#	TNumber	Код подразделения
Dep		TDep	Название подразделения
Info		TInfo	Обязанности подразделения

Таблица 2.5 – Структура таблицы «Входящая корреспонденция» (INCORS)

Поле	Ключ	Тип	Описание поля
ID	#	TNumber	Код документа
Data1		TDate	Дата получения документа
Sender		TSender	Отправитель
NumVH		TNumVH	Номер входящего документа
Data2		TDate	Дата входящего документа
Text		TCaseText	Короткое содержание документа
ID_Executor		TExecuter	Исполнитель
Info		TInfo	Отметка о контроле

Таблица 2.6 – Структура таблицы «Типы дел» (TYPECASES)

Поле	Ключ	Тип	Описание поля
Number	#	TNumber	Код
Dep	(#)	TDepNum	Код подразделения
Ind	(#)	TIndexCase	Индекс дела
CountPart		TCount	Количество томов
NameSase		TNameCases	Заглавие дела
Period		TSave	Срок хранения
PerNum		TPerNum	Номер статьи
Info		TInfo	Примечание

Таблица 2.7 – Структура таблицы «Дела» (CASES)

Поле	Ключ	Тип	Описание поля
ID	#	TNumber	Код
Case_Num		TCaseNum	Номер дела
Dep	(#)	TDepNum	Код подразделения
Ind	(#)	TIndexCase	Индекс дела
Case_Date		TDate	Дата формирования
Case_Text		TCaseText	Короткое содержание

Таблица 2.8 – Структура таблицы «Исходящая корреспонденция» (OUTCORS)

Поле	Ключ	Тип	Описание поля
ID	#	TNumber	Код документа
Data1		TDate	Дата отправления
SendTo		TSender	Адресат
Text		TSpravaText	Короткое содержание документа
Dep	(#)	TDepNum	Код подразделения, где находится копия

Таблица 2.9 – Структура таблицы «Исполнители» (EXECUTORS)

Поле	Ключ	Тип	Описание поля
ID	#	TNumber	Код документа
Data1		TDate	Дата отправления
SendTo		TSender	Адресат
Text		TSpravaText	Короткое содержание документа
Dep	(#)	TDepNum	Код подразделения, где находится копия

В спроектированной БД учитываются требования нормализации БД: поля неделимы, отсутствует дублирование информации, указаны ключи, однозначно определяющие запись и которые не являются избыточными, то есть значения ключевых полей уникальны для каждой записи.

Базы данных InterBase не имеют специальных типов для реализации полей-счетчиков. Поэтому автоматическая инициализация значений ключевых полей перечисленных таблиц осуществляется с помощью специальных *генераторов*, которые задействуются внутренними процедурами таблиц БД – *триггерами*.

Далее в таблице 2.10 перечислены задействованные генераторы и соответствующие триггеры. Начальные значения генераторов установлены в 0.

Таблица 2.10 – Список генераторов БД

Генератор	Триггер	Таблица	Изменяемое поле
Gen_Departs_ID	Departs_BI	Departs	ID
Gen_InCors_ID	InCor_BI	InCors	ID
Gen_Decrees_ID	Decrees_BI	Decrees	ID
Gen_OutCors_ID	OutCors_BI	OutCors	ID
Gen_Cases_ID	Cases_ID	Cases	ID
Gen_TypeCases_ID	TypeCases_ID	TypeCases	ID
Gen_Executors_ID	Executors_BI	Executors	ID

Отметим, что все триггеры выполняются перед добавлением записей в соответствующие таблицы.

Для оптимизации поиска документов требуемые поля проиндексированы.

2.4 Разработка структуры программного обеспечения

Взаимодействие пользователя с системой производится в диалоговом режиме. Основным соединительным элементом разрабатываемой АИС есть система меню, которая состоит из главного меню и подменю.

Функциональное меню автоматизированной системы представлено на рис. 2.5.

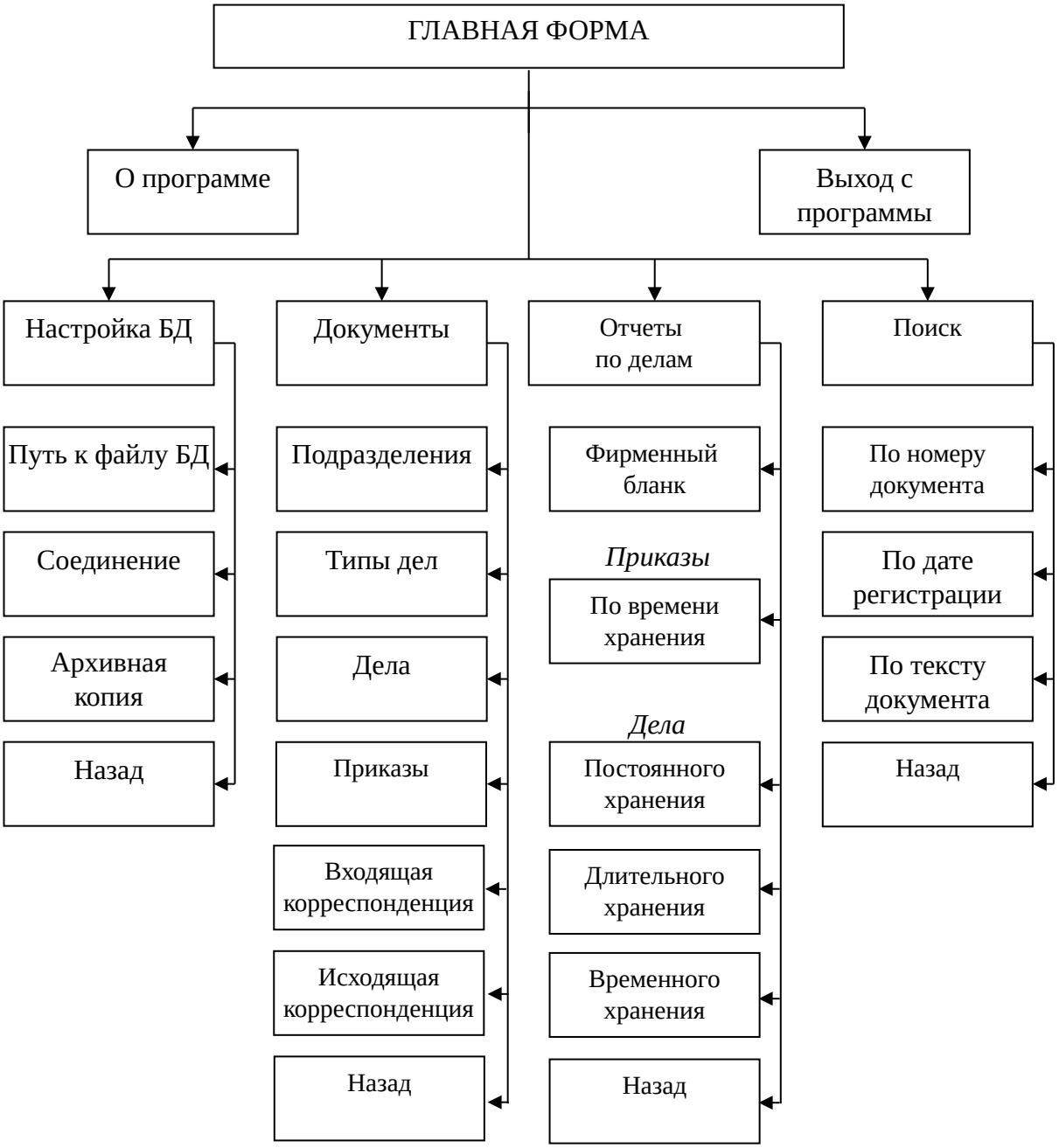


Рисунок 2.5 – Функциональное меню автоматизированной системы

Технология внутримашинной организации определяется последовательностью реализованных процедур, схем взаимосвязей программных модулей и информационных массивов.

Указанные схемы представляют из себя декомпозицию общего процесса решения задачи на отдельные процедуры преобразования массивов, называемых модулями (ввод, контроль, перезапись информации с одного МН на другой, сортировка, сжатие данных, редактирование, вывод на печать, удаление и т.д.).

Основное предназначение разрабатываемой АИС – автоматизация документооборота в образовательном заведении.

Укрупненную функциональную схему программы можно представить следующими основными блоками (рис 2.6):

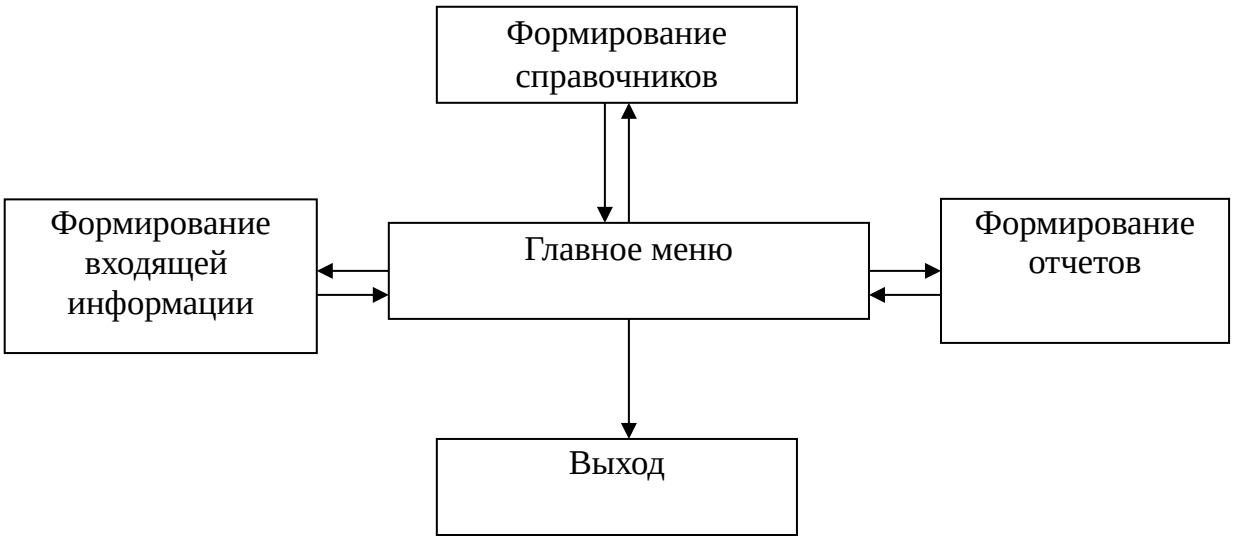


Рисунок 2.6 – Функциональная схема программы

Текст программы объединяет несколько программных модулей, описание которых представлено в таблице 3.11.

Таблица 2.11 – Программные модули системы

Модуль	Форма	Предназначение
UMain	FMain	Главный модуль. Реализация основных функций
USettings	FSettings	Инициализация начальных данных
UAccess	FAccess	Авторизация. Вход в систему
UExecutor	FExecutor	Справочник «Исполнители»
UContent	FContent	Вывод информации о программе
UAbout	FAbout	Вывод информации о разработчике

Схема межмодульного взаимодействия представлено на рис. 2.7.

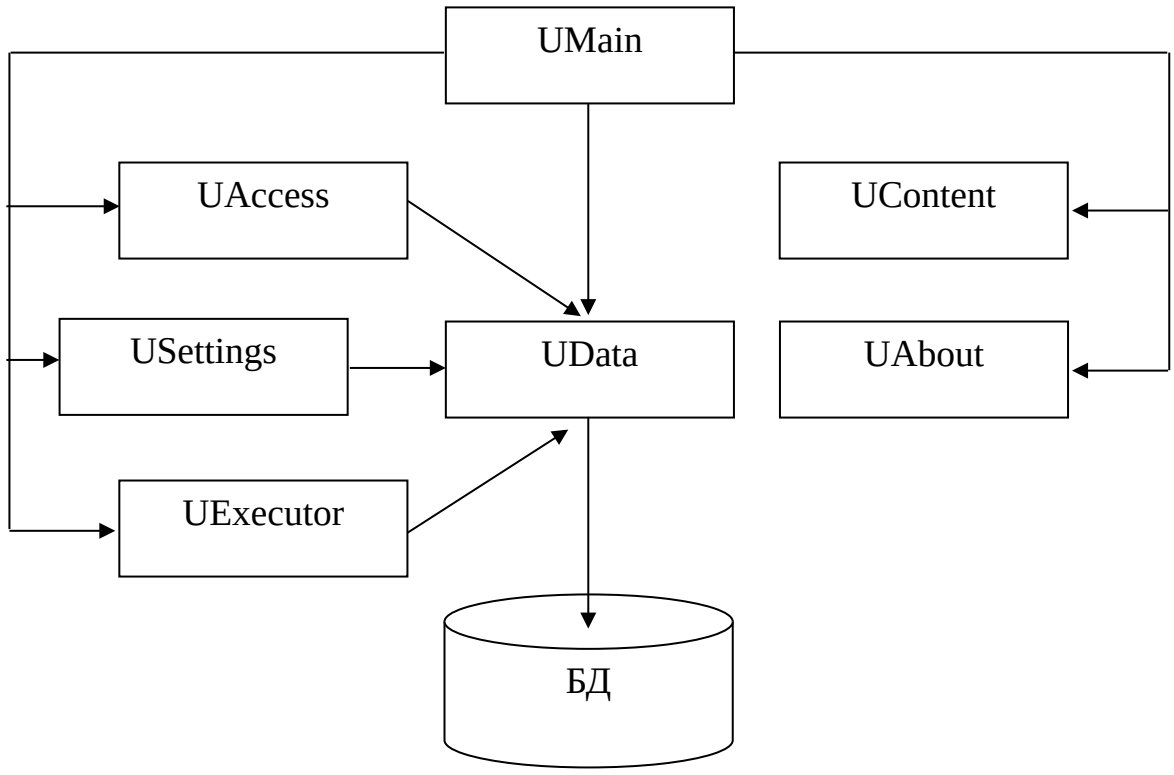


Рисунок 2.7 – Схема взаимодействия модулей программы

2.4 Разработка алгоритмов работы программного обеспечения

Работа с программой начинается с подключения клиентского приложения к БД. При этом пользователь с помощью стандартного диалога выбора может самостоятельно определить путь к файлу БД и зафиксировать его.

Автоматически с реестра системы загружаются данные о руководстве заведения, службе деловодства, о почтовом сервере. При необходимости эту информацию можно изменить.

Как указано выше, работа программы осуществляется в диалоговом и пошаговом режимах, при этом под диалогом понимается предоставление пользователю нескольких альтернатив и обработка его выбора.

Под событиями понимаются процессы, которые активизируются пользователем (например – нажатие функциональных клавиш), а также программные события – получение определенным полем фокуса редактирование или потеря фокуса ввода. На основании данных событий активизируются процедуры контроля допустимости данных.

Для понимания программной логики работы созданного приложения рассмотрим используемые глобальные переменные, процедуры, а также некоторые алгоритмы реализации функционала.

Глобальные переменные хранят значения на протяжении всего сеанса работы и доступны из других модулей.

Таблица 2.12 – Глобальные переменные

Имя	Тип	Предназначение
PathProg	string	путь к папке приложения
PathDB	string	путь к БД приложения
tn	Array[1..1000] of integer	хранение идентификаторов записей с выборки
tn2, tn3	Array[0..100] of string	хранение промежуточных данных с выборки

Продолжение таблицы 2.12

Имя	Тип	Предназначение
pwd	string	текущий пароль пользователя
DirectorName	string	Фамилия, отчество директора
Delovod	string	Фамилия, отчество деловода
SMTPSer	string	SMTP-сервер
Login	string	логин почтового ящика
Pass	string	пароль почтового ящика
A	integer	флаг результата авторизации

Основные расчеты и визуализация данных выполняются пользовательскими процедурами и функциями, представленными ниже:

function TFMain.decoder(s:string):string – декодирование текстовой строки;

function TFMain.coder(s:string):string – кодирование текстовой строки;

procedure TFMain.SGZClear – очистка ячеек компонента TStringGrid;

procedure TFMain.Word1(s:string;a, b:OleVariant; var vstart, vend: OleVariant) – определение вхождения указанной строки в документ Word;

procedure TFMain.WordInsertText(Text: string) – вставка указанного текста в документ Word;

procedure TFMain.WordGotoBookmark(Bookmark: string) – переход в документе Word на указанную закладку;

procedure TFMain.WordInit(FileName:OleVariant) – инициализация сервер MS Word;

procedure TFMain.WordEnd(FileName:OleVariant) – отключение от сервера MS Word;

procedure TFMain.TableF(U:integer) – форматирование таблицы в документе Word;

procedure TFMain.Open1Click(Sender: TObject) – вход в систему, подключение к БД;

procedure TFMain.Close1Click(Sender: TObject) – завершение работы.

В разрабатываемой информационной системе для ввода, просмотра и распечатки отчетов с информацией базы данных применяются формы. Обработка таблиц осуществляется с помощью SQL-запросов, которые формируются и выполняются компонентом IBQuery1.

Далее представлен формат SQL-запросов для осуществления основных действий над БД.

Таблица 2.13 – SQL-запросы обработки данных

Операция	SQL-команда
Добавление	insert into DEPARTS (Dep, Info) values ('*Подразделение 1*', '-----')
Редактирование	update DEPARTS set Dep = '*Подразделение 1*', Info = '-----' where ID=1
Выборка	select * from TYPECASES order by Dep, Ind

Блок-схема алгоритма обработки данных представлена в приложении Б.

Контроль достоверности информации в программе осуществляется на разных уровнях и на разных стадиях ее ввода. Он призван обеспечить отсутствие ошибок при вводе тех или иных данных, поддержку достоверности информации в процессе функционирования СУБД.

К средствам контроля относятся:

– задания входных данных, максимальных и минимальных значений, шаблонов для элементов ввода;

– контроль изменения информации при работе с ней и сообщение об изменении информации перед сохранением;

– использование кэширования данных;

– использование целостности связей;

– использование таблиц-справочников;

– ведение копий таблиц;

– отражение в справочной системе правил ввода информации;

– разграничение доступа пользователей к информации, фиксация пользователя, вводящего информацию.

Степень контроля информации в значительной степени зависит от степени важности используемой СУБД и от последствий, к которым могут привести ошибки. Слишком жесткий контроль может осложнит функционирование программного продукта, не дав ощутимого эффекта.

При разработке программного комплекса использован спектр средств, обеспечивающих программный контроль входной информации.

Так на уровне программного кода для ввода даты поступления или отправки документа, названия отправителя или адресата применен шаблон ввода, контролирующий также обязательность введения. Данные о подразделениях и индексы дел вводятся путем выбора из соответствующих справочников, при этом в больших справочниках интегрирована система поиска.

Перед выполнением операций изменения записей БД система проверяет корректность входных данных и требует подтверждения проведения операции.

При вводе данных у соответствующие компоненты, автоматически проверяется корректность их представления. Например, ввод числового значения сопровождается игнорированием всех символов, кроме цифр, десятичной точки и клавиши BACKSPACE. Также блокируется попытка повторного ввода десятичного разделителя. Указанные возможности

реализуются событием *KeyPress* для компонентов ввода, текст кода которого приведен ниже:

```
var vrPos,vrLength,vrSelStart : byte;
const I : byte=1;                //I+1 – количество знаков
                                        после точки
Begin

with Sender as TEdit do
begin
   vrLength:=Length(Text);      //определение длины текста
   vrPos:=AnsiPos('.', Text);   //проверка наличия точки
   vrSelStart:=SelStart;        //определение положения
                                  курсора
end;

case Key of
'0'..'9' :                      //проверка положения курсора и
                                количества знаков после точки

if (vrPos>0)and(vrLength-vrPos>I) and (vrSelStart>=vrPos)
then Key:=#0;                   //"погасить" клавишу

',', '.' :                      //блокировка повторного ввода
                                  точки
if (vrPos>0) or (vrSelStart=0) or (vrLength=0)
then Key:=#0
else Key:=’.’;                   //замена запятой на точку

#8 : ;                          //реакция на клавишу
                                  'BackSpace’'

else Key := #0;                 //блокировка запрещенных
                                  символов
end;

End;
```

Также на уровне БД при формировании структуры таблиц определены поля, для которых при отсутствии ввода данных, определяется значение по умолчанию (например 0 для COUNT), или проверяется уникальность содержимого (ID, DEP).

Правильные связи между взаимосвязанными таблицами БД позволяют автоматически устанавливать и поддерживать организацию целостности связей (таблицы OUTCORS - DEPARTS т.д.).

Для запрета корректировки данных некоторых параметров использовано свойство *ReadOnly* для отдельных столбцов (DBGRID).

Ошибка в информации может возникнуть не только в результате ввода некорректных данных, но и в случае отсутствия сохранения кэшированной информации в соответствующих таблицах БД. По этой причине предусмотрен вывод сообщений об имеющихся изменений, или об намерении удаления записи. Для этого используются соответствующие стандартные диалоговые окна. Удаление же данных из таблиц физически доступно только одному человеку – администратор.

Корректность фиксации данных в БД также достигается за счет использования механизма транзакций – манипуляции с БД, которая переводит ее из одного целостного состояния в другое. Если одно из изменений, которое вносится в БД в рамках транзакции, завершается неудачно, производится откат до состояния БД, имевшего место до начала транзакции, то есть все изменения, внесенные в БД в рамках транзакции, или одновременно подтверждаются, или не подтверждается ни одно из них.

Если в рамках транзакции произошел сбой при выполнении одной из операций, автоматически отменяется результат выполнения всех других операций, иначе информация в БД будет недостоверной. Например, если из таблицы "ПОДРАЗДЕЛЕНИЯ" изъять запись, и при этом не выполнять никаких манипуляций над таблицей исходящей корреспонденции, будем иметь "зависшие" данные в последней таблице.

2.5 Руководство пользователя программного обеспечения

2.5.1 Установка приложения на ПК

Установка приложения производится простым копированием дистрибютивной папки ZAUCH с носителя на жесткий диск ПК.

Ниже приведена файловая структура дистрибютива:

PZauch.exe	–	исполняемый файл программы
Zauch.ini	–	файл, содержащий путь к БД системы и данные начальной инициализации;
Zauch.fdb	–	файл базы данных;
content.rtf	–	текст справки
blank.doc	–	шаблон фирменного бланка
dep.doc	–	шаблон бланка отчета по подразделениям
cases.doc	–	шаблон бланка дела
type_case.doc	–	шаблон бланка отчета по типам дел
rep1.doc	–	шаблон общего отчета по документам
rep2.doc	–	шаблон отчета о категориях и количестве дел
rep3.doc	–	шаблон отчета по делам

Предварительно, необходимо установить и загрузить сервер БД Firebird. Скачать дистрибютив указанного сервера БД можно по адресу: http://www.firebirdsql.org/en/downloads/

2.5.2 Настройка подключения к БД и вход в систему

При первом запуске программы следует настроить связь с БД. Подпункт “Настройка” пункта “Файл” главного меню программы активизирует форму подключения к БД (рис. 3.4).

Путь к серверной БД формируется через использование сетевого протокола, формат строки TCP/IP которого следующий:

<HOST>:<Local Path\Zauch.gdb>, (3.1)

где HOST – имя хоста (сервера),

Local Path – локальный путь к БД в файловой системе.

Например, COMP1:D:\Zauch\Zauch.fdb

Имя хоста (IP-адрес сервера) можно получить у администратора сети.

Иногда между именем сервера и файловым путем к БД следует ставить ‘\’.

Например, COMP1**:**\D:\Zauch\Zauch.fdb

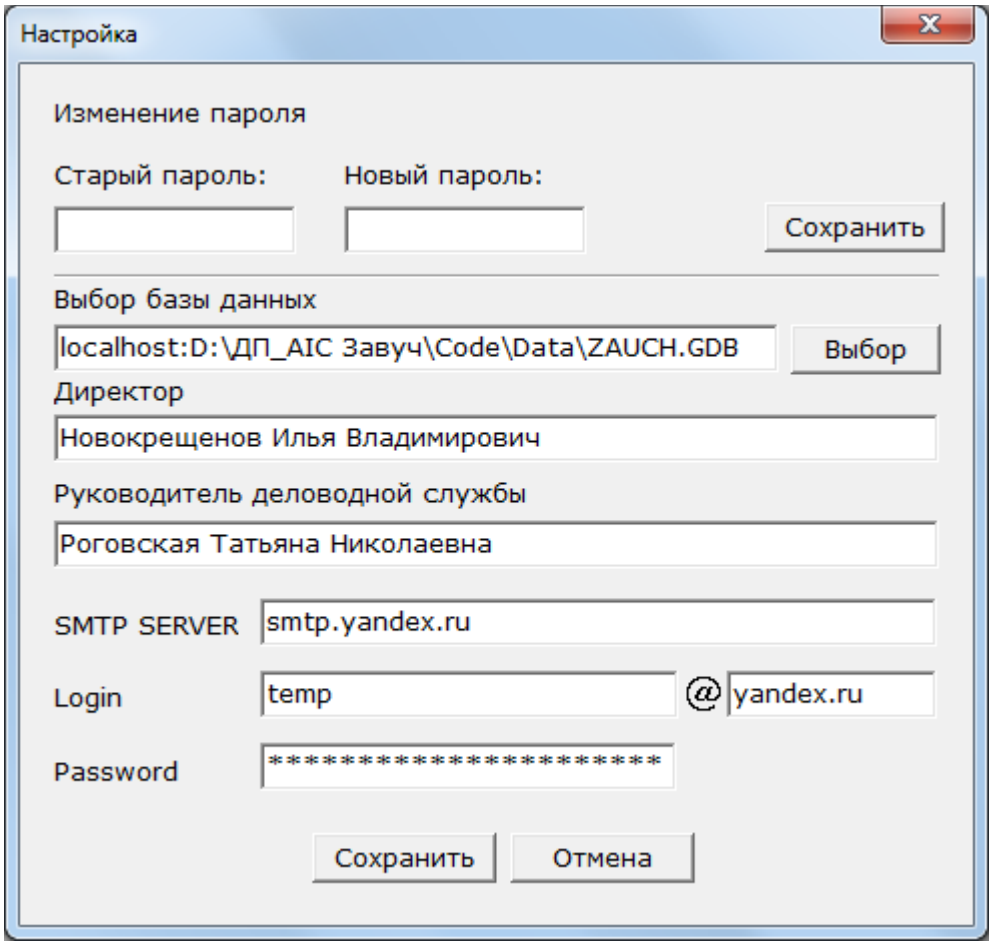


Рисунок 2.8 – Окно начальной инициализации

Если С и БД установлены на одном ПК, то в качестве сетевого имени хоста используют локальный адрес («заглушка») localhost с IP-адресом 127.0.0.1.

При наличии сети из двух машин, соединенных кросовым кабелем, то можна выбрать любой IP-адрес для сервера (кроме 127.0.0.1) и IP-адрес для клиентского ПК.

Для ускорения соединения с БД в файл **HOSTS** необходимо внести следующую запись: <host> <IP-адреса>. *Например*, localhost 127.0.0.1

В Windows NT2000 файл HOSTS размещается в C: \WINNT\system32\drivers\etc\, а в Windows 95/98/ME/XP/Server2003 – в C: \Windows.

Строку TCP/IP соединения можно построить, вызвав соответствующее диалоговое окно:

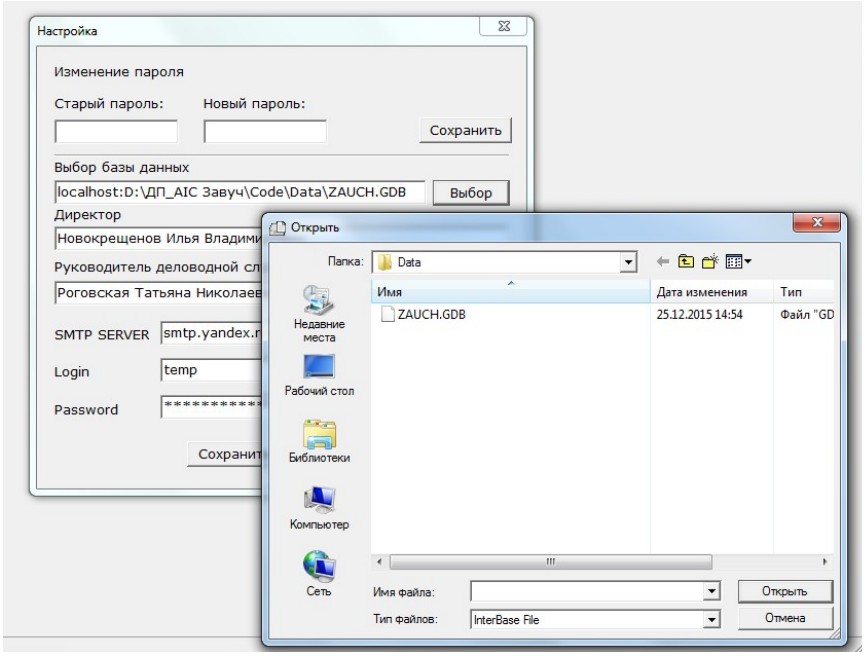


Рисунок 2.9 – Окно соединения с файлом БД

После правильного ввода пути к БД, его следует сохранить с помощью кнопки “Сохранить”. Путь к БД сохраняется в файле Zauch.ini в текущем каталоге программы.

Система защищена программным образом и ее активизация происходит после ввода пароля в соответствующее поле:



Рисунок 2.10 – Окно входа в программу

2.5.3 Работа в среде АИС "Учет документов"

Взаимодействие пользователя с системой производится в диалоговом режиме, то есть разработанная система есть меню-ориентированной.

Внешний вид главной формы программы приведен на рис. 2.11.

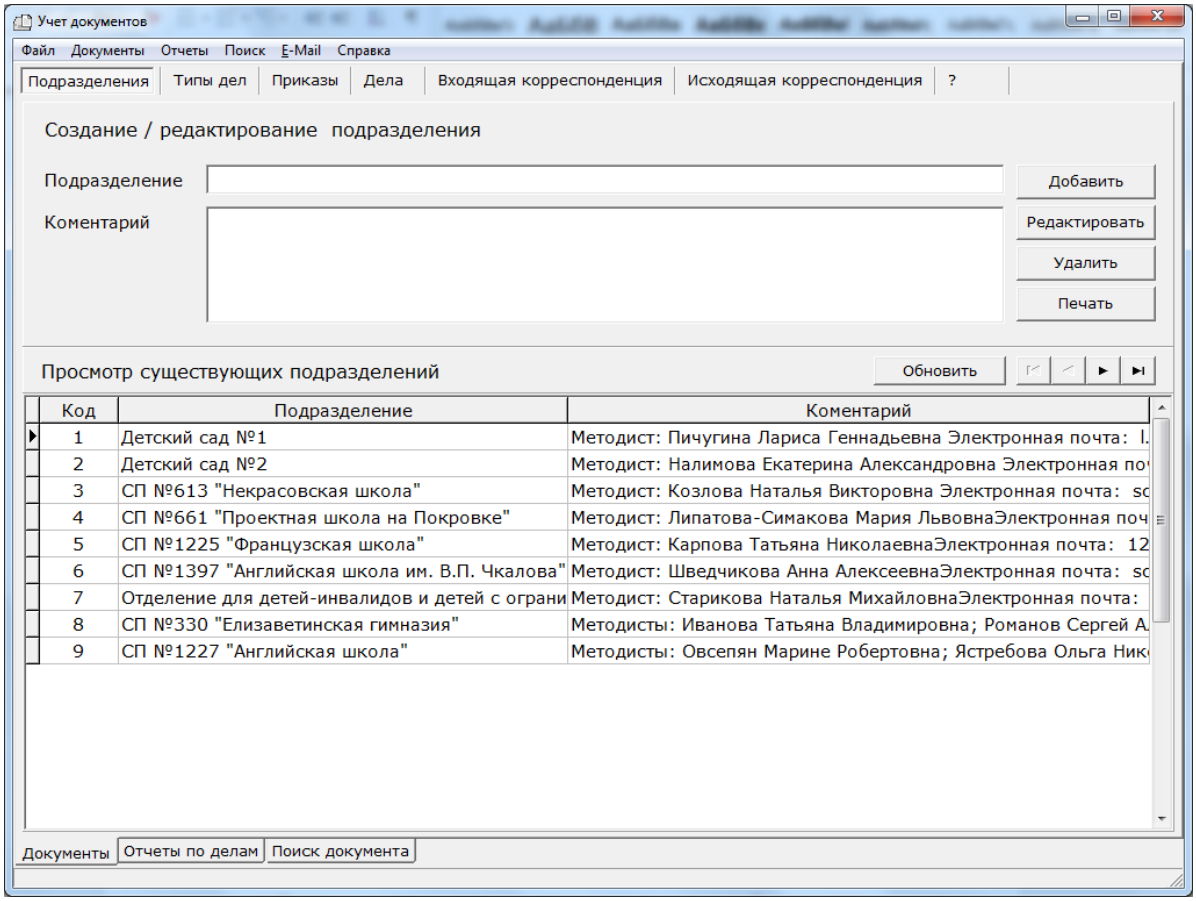


Рисунок 2.11 – Главная форма программы. Раздел “Документы”

Заметим, что при активизации пунктов главного меню и переходе к работе в выбранном режиме, система переходит в модальный состояние, то есть до окончания обработки текущих данных доступ к другим пунктам меню блокируется.

Над записями в таблицах можно проводить стандартные операции: добавление записи, редактирования, скроллинг. Попытки внесения изменений в базу данных необходимо подтверждать в стандартных диалоговых окнах.

Программное приложение имеет достаточный набор различных отчетов, список которых активизируется соответствующим пунктом меню. В левой части вкладки "Отчеты по делам" приведен перечень отчетов, большую часть занимает таблица, содержащая отобранные документы (рис. 2.12).

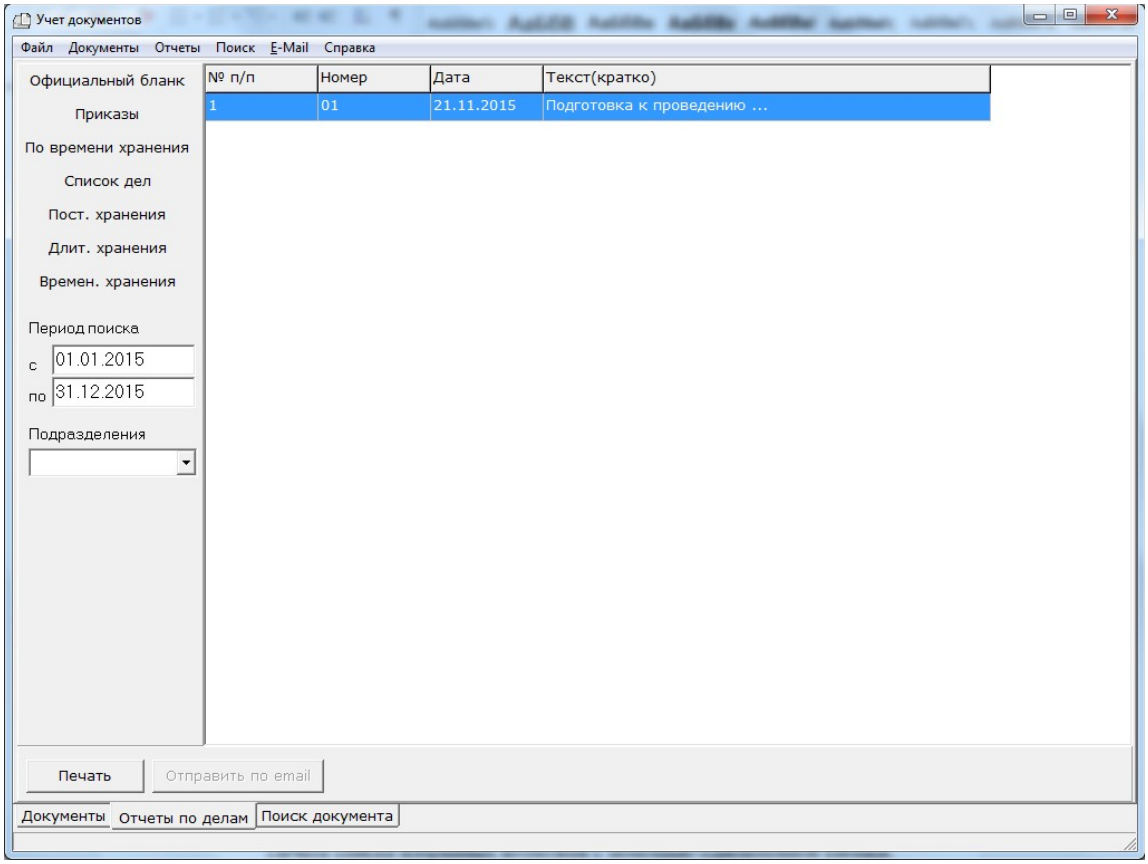


Рисунок 2.12 – Главная форма программы. Раздел “Отчеты по делам”

Отбор документов осуществляется согласно указанного периода поиска. Через двойной щелчок вызывается форма с полной информацией о выбранной записи.

Печать списка избранных возможен с помощью одноименной кнопки.

Для поиска и печати отдельных документов следует перейти к соответствующей вкладке "Поиск документов" (рис. 2.13).

Параметрами поиска есть номер, дата или текст документов. Активизация поиска осуществляется кнопкой "Поиск".

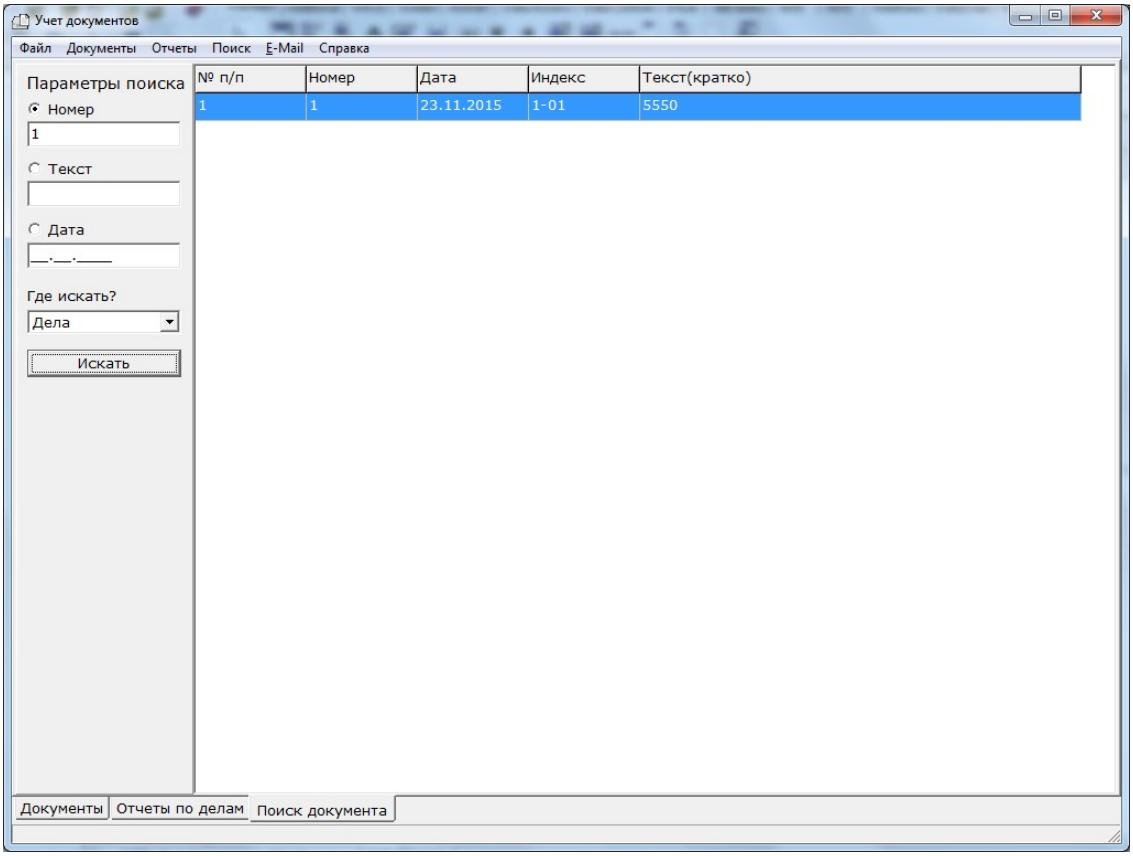


Рисунок 2.13 – Главная форма программы. Раздел “Поиск документов”

Информация о программе вызывается после нажатия горячей клавиши "F1". Завершается работа с программным приложением после выбора подпункта "Завершить" пункта «Файл» главного меню.

2.5.4 Архивное копирования БД

Корректное резервное копирование базы данных Firebird (InterBase) компактно сохраняет мета данные и данные БД на жестком диске. Рекомендуется перезаписывать файлы резервных копий на сменные носители и хранить в физически защищенном месте вне сервера.

Регулярное резервное копирование является основой для поддержания БД в хорошем состоянии, обеспечения целостности данных и их безопасности. Утилита командной строки резервного копирования и восстановления данных в

FireBird gbak создает и восстанавливает резервные копии БД. Она создает независимый от платформы архив фиксированного «фото» БД на момент сохранения.

Для вызова утилиты gbak следует в командной строке вести следующую команду:

gbak -b Zauch.fdb Zauch.fdb

Для обеспечения удобства пользователя в программе автоматизирован архивное копирования БД. Для этого следует активизировать подпункт "Архивная копия" пункта «Файл» главного меню.

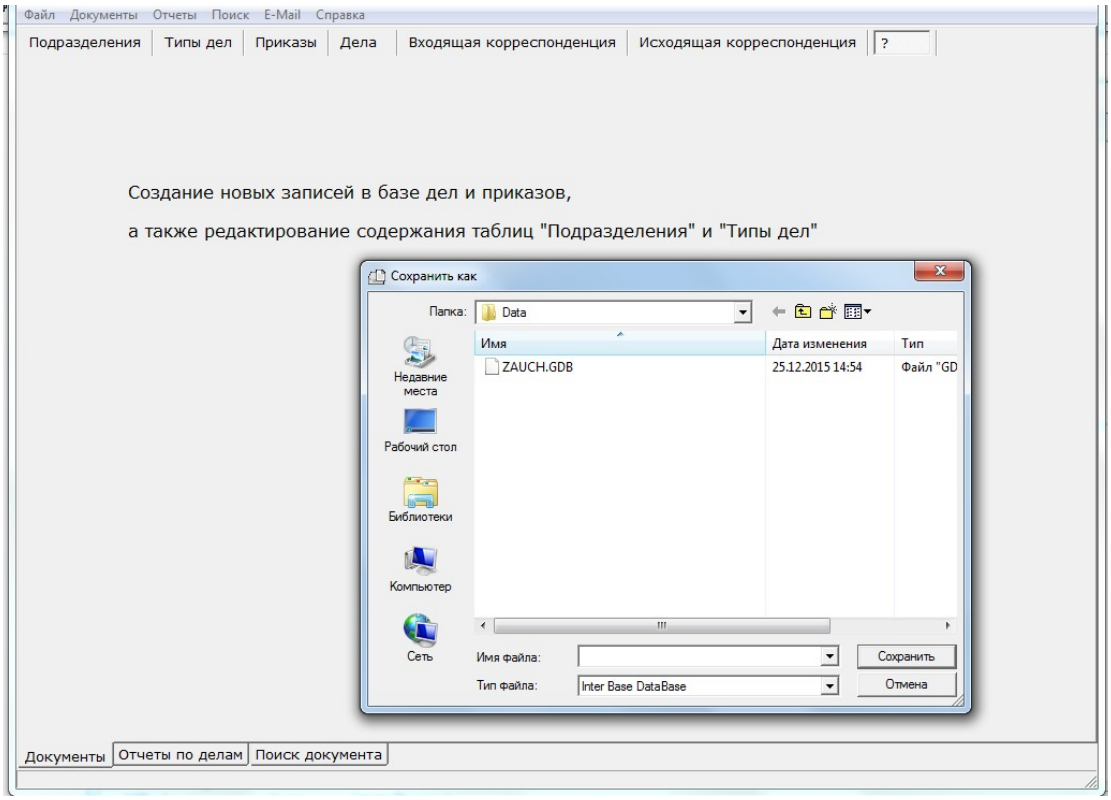


Рисунок 2.14 – Архивное сохранение БД

2.5.5 Возможные проблемы и их устранение

Причинами невозможности доступа к БД могут быть:

– проблемы с сетью

– неактивность сервера БД или некорректность его установки;

– некорректно указан путь к БД.

Проверка запуска сервера БД осуществляется через запуск менеджера сервера БД с панели управления Windows

При этом на ПК не должно быть FireWall-ов или прокси, которые не дают ему слушать порт +3050 по TCP/IP (или firewall/прокси нужно настроить, чтобы не мешали Interbase).

После этого нужно убедиться, что если у вас в сети нет серверов Netware, то не один клиент или сервер не имеет установленной поддержки протоколов IPX/SPX (NWLink и т.п). Этот протокол только мешает работе.

Вообще, если ОС установлена корректно, и сеть правильно настроена, то InterBase или Firebird должны также функционировать нормально. По поводу настройки сети следует обращаться к ее администратору или к программной документации ОС.

Ошибка «unavailable database» может генерироваться при попытке присоединиться к БД через "локальный протокол", например, d: \Zauch\Zauch.fdb. Вместо этого необходимо использовать сетевой протокол, например localhost:d:\Zauch\Zauch.fdb.

Причинами ошибки могут быть следующие:

• клиентская часть (gds32.dll, fbclient.dll) не соответствует версии сервера

• клиентская часть не поддерживает локальный протокол вообще (например, в Firebird 1.5.1 for Windows, Classic). Также локальный протокол не работает, если на Windows активизированы службы Terminal Services.

• особенности конкретного логина или версии операционной системы

Во всех случаях проблем с локальным протоколом рекомендуется проверить все вышеперечисленные варианты, и если их не удалось устранить –

использовать локальный сетевой протокол для соединения с БД (например, localhost:d:\Zauch\Zauch.fdb).

После установки SP2 на Windows XP автоматически активизируется встроенный Firewall и блокирует доступ практически ко всем портам TCP/IP, в том числе и порт 3050, который используют InterBase или Firebird для работы (по умолчанию). Поэтому порт 3050 в Firewall нужно открыть (если не будет работать, необходимо еще и открыть порт 113).

3. ОБОСНОВАНИЕ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ПРОЕКТА

3.1 Выбор и обоснование методики расчета

Как любое приложение, разработка системы электронного документоборота в учреждении может быть экономически обоснована с точки зрения непосредственной прибыли, которая обычно измерима, а так же с точки зрения косвенной прибыли, которая, скорее, имеет качественную, но ничуть не менее важную ценность.

Под экономической эффективностью будем считать результативность экономической системы, которая выражается отношением полезных конечных результатов ее функционирования к затраченным ресурсам, другими словами – мера соотношения цены и затрат, а эффектом может быть результат внедрения какого-либо мероприятия выраженного в стоимостной форме.

Для определения эффективности внедрения разработки необходимо определить методику ее расчета.

Поскольку предметом защиты является разработка проекта, направленного на улучшение условий труда, то экономическую эффективность обоснуем качественно. Количественно рассчитаем только затраты, предварительно обосновав их необходимость.

Для обоснования экономической эффективности разработки проекта, как правило, используется методика расчета, в которой целесообразные варианты построения выбираются путем балансирования показателей приращения эффекта E, получаемого за счет создания или совершенствования продукта, и затрат Z.

Математически эту задачу формулируют в виде формулы (1):

$$Max\ E, \text{ при } Z=\text{Const} \qquad (1)$$

или в виде обратной задачи в виде формулы (2)

Min Z, при *E*=Const (2)

Эффективностью будем называть величину эффекта, приходящуюся на единицу производственных затрат.

Существует реальный и расчетный эффект. Расчетный эффект рассчитывается, реальный эффект можно получить сразу. При определении эффекта и реального, и расчетного рассчитывается прямой и косвенный эффект.

Основным показателем при использовании типовой методики расчета экономической эффективности создания и функционирования системы является годовая экономия текущих затрат, вычисляемая по формуле (3):

$$\Delta C_m = \Delta C_p + \Delta C_k \quad (3)$$

где ΔC_p – прямая экономия;

ΔC_k – косвенная экономия.

Поскольку учесть оба параметра не представляется возможным, определим методику расчета затрат на реализацию и внедрение ПО.

Расчет показателя затрат $Z_{ср}$ производится по формуле (4), т.е. складывается из текущих затрат:

$$\Delta Z_{ср} = Z_1 + Z_2 + Z_3 + Z_4 + Z_5 + Z_{пр} \quad (4)$$

где Z_1 – затраты на оплату труда персонала;

Z_2 – начисления на фонд оплаты труда (в настоящее время это социальный налог 26% от ФОТ);

Z_3 – затраты на разработку, внедрение и эксплуатацию программного приложения;

Z_4 – амортизация оборудования, как правило рассматривается линейная со сроком службы от 3 до 5 лет;

Z_5 – прочие затраты (расходы на командировки, информационные расходы, плата за кредит, налоги, представительские расходы). Включает также затраты, связанные с потерей времени от момента получения заказа до момента начала работы по ней, выраженные в руб.

$Z_{пр}$ – предпроизводственные затраты, которые требуются дополнительно.

Предпроизводственные затраты – это затраты на создание (покупку) программных средств, на обучение специалистов и т.д.

Дополнительные капиталовложения *КД*, необходимы для создания системы. К ним относятся: строительство, ремонт, аренда помещения, покупка сетевого оборудования, средств связи и др. Стоимость капитальных вложений приводятся к одному году функционирования системы, срок функционирования системы оговаривается.

Срок окупаемости дополнительных капитальных вложений, вычисляемый по формуле (5):

$$T = КД / \Delta C_m \qquad (5)$$

Расчетный коэффициент эффективности дополнительных капитальных вложений E_p является величиной, обратной T, то есть $E_p = 1/T$.

$E_н$ – нормативный коэффициент эффективности.

Если расчетный коэффициент превышает или равен $E_н$, то есть если $E_p \geq 0.33$, то проект принимается к внедрению и создание разработки эффективно.

3.2 Расчет затрат и окупаемости реализации проекта

Затраты на разработку и внедрение ПО складываются из следующих основных частей:

- приобретение аппаратного и программного обеспечения;
- расходы на непосредственную разработку. Включают в себя стоимость разработку и дизайн программного продукта, технический консалтинг, проектирование базы данных, коддинг;
- краткосрочный штат разработки;
- внедрение (самый важный и самый сложный процесс, включающий в себя бизнес-анализ и доработку системы под конкретные требования заказчика).

В затратную часть создания СЭД относятся такие расходы как: стоимость ИСР Delphi 6, расходы на электроэнергию, заработная плата программиста, расходы на канцелярские товары и расходные материалы для компьютера, аренда помещения, амортизации компьютеров и оргтехники и прочие расходы.

Дистрибутив программы размещается на оптических носителях информации. Расходы по созданию ПО указаны в таблице 3.1.

Таблица 3.1. Данные затрат разработки и внедрения ЭУ

Затраты, связанные с разработкой, внедрением и эксплуатацией	Сумма
Установка и конфигурация необходимого ПО	300
Разработка структуры приложения	800
Проектирование БД и логики на стороне БД	1 000
Разработка модулей управления данными	2 500
Разработка логики поиска и фильтрации	600
Разработка отчетов	600
Тестирование и отладка	1 500
Итого за разработку и внедрение системы	7 300

Поскольку разработка программного обеспечения в нашем случае некоммерческая, то стоимость пакета ИСР Delphi 6 Personal составляет $99 или 5 800 руб. Таким образом, $Ц_{по}$ = 5 800 руб

Время реализации проекта составило 6 дней. Рассчитаем затраты на использование электроэнергии для восьмичасового рабочего дня.

Таблица 3.2. Данные по использованию электроэнергии

Наименование	кол-во	Мощность, кВт	Потребление, кВт час
Компьютер	1	0.4	0.4
Освещение	3	0.1	0.3
Итого			0.7

Тариф на электроэнергию: 1 кВт / ч = 5.03 руб. Тогда, затраты на электроэнергию составят: 6 × 8 ×0.7 × 5.03 = 169 руб.

Таким образом, затраты на разработку и внедрение программного продукта составили: 5 800 + 7 300 + 169 = 13 269 руб.

Так как помещение и оборудование уже имеется в наличии, рассчитаем годовую сумму амортизационных отчислений.

Годовая сумма аморт. отчислений рассчитывается по формуле:

$$A = Ф \times N_A / 100\%, \quad (6)$$

где Ф – первоначальная стоимость основных фондов по видам, руб.;

N_A – норма амортизации по видам основных фондов, в %.

Годовую сумму амортизационных отчислений отразим в таблице 4.3

Таблица 3.3. Расчет годовой суммы амортизационных отчислений

Элементы основных фондов	Количе-ство	Стоимость, руб.	Сумма руб.	Норма амортиза-ции, %	Амортизацион-ные отчисления, руб.
Компьютер	1	26 400	26 400	20%	5 280
Помещение	14.2 $м^2$	5 500	78 100	3%	2 343
Итого:					7 623

В результате, годовая сумма амортизационных отчислений составляет 7 623 руб.

Рассчитаем сумму амортизационных отчислений для перечисленной группы оборудования с учетом числа календарных дней на разработку программного обеспечения по формуле:

$$A_{факт} = A_{год} \times T_{факт} / 365 \quad (7)$$

Исходя из того, что трудоёмкость создания информационной системы составляет 6 дней, а коэффициент использование ПК – 0.9, рассчитываем амортизацию оборудования за этот период:

$A = 7\ 623 \times 6 \times 0.9 / 365 = 112.78$ руб.

Затраты на заработную плату, включаемые в себестоимость программы составили 7 300 руб. Отчисления на социальное страхование составили:

$ECH = ЗП_{пр} \times 26\% = 1\ 898$ руб.

Рассчитаем себестоимость программного продукта по формуле 4:

$C_{ст} = З_{пр} + ЗП_{пр} + ECH + A = 7\ 300 + 169 + 1\ 898 + 112.78$

$C_{ст} = 9\ 479.78$ рублей.

Полученная себестоимость является приблизительной, поскольку в ней не учтены некоторые детали, которые существенно не влияют на итог.

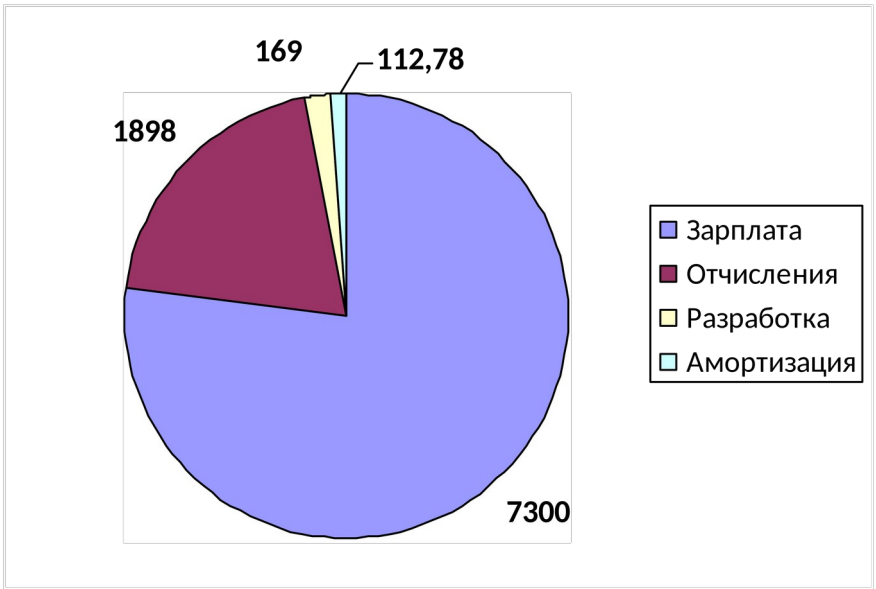


Рисунок 3.1. Распределение затрат на разработку продукта

Исходя из нормального уровня рентабельности 20% мы можем определить цену разработанной нами программы:

$$Ц = C_{ст} \times (1 + R/100\%), \quad (8)$$

где $C_{ст}$ - себестоимость разработки программы;

R - планируемый уровень рентабельности.

Таким образом, цена разработки и внедрения системы документооборота составляет:

Ц = 9 479.78 × (1 + 0.2) = 11 375.78 руб.

3.3 Расчет экономических результатов внедрения АРМ

Эффективность автоматизации рабочего места обусловлена действием ряда факторов организационного, информационного и экономического характера.

Организационный эффект проявляется в освобождение работников от рутинных операций по систематизации и группировке учетных данных, сверки показателей. Экономический фактор проявляется в том, что учетная информация, имеющая целью полное и своевременное отражение и состояние объекта и причин, влияющих на его развитие, в конечном счете, направлена на улучшение использование ресурсов

Рассчитаем срок окупаемости и экономию затрат от внедрения программного продукта.

Годовая экономия на себестоимости продукции объекта управления;

$$\Delta C_{себ} = \Delta C_{к} + \Delta C_{врем},$$

где

$\Delta C_{к}$ – экономия на канцелярии;

$\Delta C_{врем}$ – сокращение времени заполнения документов.

Общая сумма потерь за год, вызванная недостатками ведения документации, равна примерно 26 тыс.руб. Система позволит снизить эти потери на 70% в год. Поэтому ΔШ = 18 200 руб.

Учитывая, что $\Delta C_к$ =1500 рублей в год и ΔШ =18 200 рублей в год получаем:

$Э_{косв}$ =$\Delta C_{себ}$ +ΔШ =1 500 +18 200=19 700 руб./год

$\Delta Э_{год} = Э_{косв} + Э_{прям}$ = 19 700 + 0=19 700 руб./год

$Э = \Delta Э_{год} – П$ =19 700 – 11 375.78 = 8 324.22> 0.

Годовой экономический эффект представляет собой абсолютный показатель эффективности. Система считается эффективной, если $Э > 0$. Рассчитаем время окупаемости от внедрения ПО:

1. Расчетная прибыльность (рентабельность):

$E_р = \Delta Э_{год} / К$ =8 324.22 / 9 479.78 =0.88

1. Срок окупаемости:

$Т_{ок} = 1 / E_р = К / \Delta Э_{год}$ = 1,14 года

Таким образом, срок окупаемости проекта составляет – 13,7 месяцев.

Приведенный расчет показал, что общие затраты на разработку системы электронного документооборота силами учреждения окупаются в денежном и социальном факторах.

4. ЭКОЛОГИЯ И БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

Охрана труда – это система законодательных актов, а также предупредительных и регламентирующих социально-экономических, организационных, технических, гигиенических и лечебно-профилактических мероприятий, средств и методов, направленных на обеспечение безопасных условий труда.

Основной задачей охраны труда является уменьшение до минимума вероятности получения травм, или заболевания работающих, обеспечение комфортных условий труда. Практически невозможно построить производственный процесс с полной гарантией безопасности и безвредности.

Операторы персональных ЭВМ, как правило, к концу трудового дня могут ощущать боль, усталость в глазах, головные боли, боли в областях шеи, спины, рук. Очевидно, что длительное воздействие вредных факторов ЭВМ могут привести к хроническим мигреням, к потере зрения, стрессовым состояниям, нарушению осанки и сколиозу, неврологическим заболеваниям, что соответственно чревато снижением работоспособности или неспособностью продолжать работу при таких условиях труда [3].

Источником проблем, связанных со здоровьем людей, работающих с ПК, может быть дисплей с электронно-лучевой трубкой. На сегодня благодаря новым технологиям и достижениям прогресса процент используемых CRT-дисплеев сравнительно, но дисплеи на жидких кристаллах или других технологиях оказывают также неблагоприятное воздействие на здоровье человека.

Работа с ПК, как и любые другие производственные процессы сопряжена с опасными и/или вредными факторами.

Опасные производственные факторы воздействуют на работающего и в определенных условиях приводят к травме, острому отравлению или другому

внезапному резкому ухудшению здоровья, в крайних случаях – к летальному исходу.

Вредный факторы могут привести к заболеванию, снижению работоспособности и(или) отрицательному влиянию на здоровье потомства. В зависимости от характеристики (уровня, концентрации и др.) и продолжительности воздействия вредный производственный фактор может стать опасным.

Питание большинства ПК осуществляется от сети 220В. Такое напряжение – опасное для человека, поэтому при использовании ПК существует риск поражения электрическим током, последствием которого являются ожоги, сбой в работе сердечной мышцы, нервной системы, потеря сознания.

Высоковольтный блок строчной развертки дисплея с электронно-лучевой трубкой вырабатывает высокое напряжение, порядка 25кВ. Работа таких дисплеев сопровождается возникновением мягкого рентгеновского излучения, которое является вредным фактором.

Во время работы электронно-лучевые устройства происходит электризация пыли и мелких частиц. Вследствие проявления статического электричества, частицы притягиваются к экрану и ухудшают видимость, что увеличивает нагрузку на глаза.

Продолжительная работа с ПК без перерывов, неправильное положение тела, монотонность труда, значительный объем информации, обрабатываемый в единицу времени приводят к напряжению зрения и внимания, к интеллектуальным, эмоциональным и длительным статическим нагрузкам. Анализ вредных и опасных факторов убеждает в необходимости защиты от них.

Таким образом, при эксплуатации ПК могут проявляться такие опасные и вредные факторы как поражение электрическим током, статическое

электричество, электромагнитное излучение, психофизические вредные и опасные факторы.

Проанализируем влияния опасных и вредных факторов на пользователя.

Воздействие при прохождении электрического тока через человека приводит к травмам, которые в свою очередь могут привести к летальному исходу.

Типы воздействия электрического тока:

- термическое – нагрев тканей, ожог тканей;
- электролитическое – разложение органических жидкостей;
- биологическое – нарушение деятельности и процессов в организме (деятельности сердца и центральной нервной системы);
- механическое – получение травмы в результате судорог.

Тяжесть поражения током зависят от его силы, длительности воздействии на человека, области воздействия, состояние окружающей среды, пола, возраста и состояния человека и других параметров.

Наэлектризованная (ионизированная) пыль, которая образуется при длительной работе дисплеев с электронно-лучевой трубкой, является причиной воспалений кожи и осложняющим фактором при заболеваниях дыхательной системы.

Электромагнитные излучения низкой частоты могут вызывать изменения в клетках живых существ. Переменное электромагнитное поле с низкими частотами вовлекает в колебательный процесс молекулы любого живого организма. Результатом длительного воздействия этого поля является изменение активности ферментов и влияет на иммунитет.

К психофизиологическим опасным и вредным факторам относятся[3]:

- напряжение органов зрения;
- интеллектуальные, эмоциональные и статические нагрузки;
- монотонность труда;

- значительный объем информации, обрабатываемый в единицу времени;
- нерациональность в организации рабочего места.

К концу рабочего дня операторы ПК могут ощущать переутомление глаз, головную боль, тянущие боли в мышцах шеи, рук и спины, снижение концентрации внимания.

Изображение на дисплее отличается от привычных глазу объектов наблюдения, поскольку изображение светится, мерцает, состоит из множества дискретных цветных точек, которые не соответствует естественным цветам. Основные проявления чрезмерной нагрузки на орган зрения являются: затуманивание зрения, искажение цветопередачи, жжение, ощущение «песка» в глазах, боли при движении глаз.

Длительная малоподвижная работа на компьютере может стать источником тяжелых заболеваний позвоночника, застоев кровотока в тазу и ногах. Постоянные статические нагрузки на кисти пальцев приводят к заболевания нервов, сухожилий и мышц руки.

Рассмотрим методы и средства защиты пользователей от воздействия на них опасных и вредных факторов.

Для создания защиты от поражения электрическим током используется технический метод – зануление.

Зануление – это соединение корпуса и металлических открытых частей прибора, которые не находится в процессе работы прибора под напряжением, с нулевым защитным проводником. Данный метод применяется в сетях с напряжением менее 1000В. Принцип данной защиты заключается в следующем: при замыкании одной из фаз на заземляющий корпус, происходит отключение прибора от сети питания. Таким образом, осуществляется защита человека от поражения электрическим током.

Защита от электромагнитных полей низкой частоты заключается в необходимости:

- соблюдения безопасное расстояние (рекомендуемое расстояние для источников электромагнитных полей не менее 0.5 *м*)

- ограничения времени работы с источником электромагнитных полей низкой частоты (рекомендуемое время не более 4 часов)

- экранирования источника (в зависимости от характеристик излучения и требованиям к безопасности применяют различные методы экранирования)

Защита от статического электричества достигается путем:

- использования нейтрализаторов статического электричества;
- создания контурного заземления;
- уменьшения или исключения синтетических тканей и покрытия;
- экранирования;
- поддержки в помещении подвижность воздуха не менее 0.2 м/с.

Персоналу и пользователям ПК и других устройств для уменьшения статического электричества и его влияния необходимо использовать спецодежду или одежду из несинтетических материалов (хлопок, лен, кожа).

Защита от вредных психофизических факторов (эргономические требования), имеющих место при эксплуатации ПК предполагает правильную организацию рабочих мест и рабочего времени оператора.

Размещение рабочих мест с использованием ПК не допускается в подвальных помещениях. На одно рабочее место должно выделяться не менее 6 $м^2$ площади и 20$м^{3.}$ объема.

В помещении, в котором есть рабочее место с ПК необходимо поддерживать следующий микроклимат:

- относительная влажность воздуха 55-65%
- подвижность воздуха не менее 0.2м/с
- температура около 20°С

Рекомендуемые требования к освещению рабочих мест при работе с ПК и документами:

- наличие естественного и искусственного освещения;
- боковое естественное освещение рабочего места, желательно слева;
- наличие регулируемых жалюзи, занавесок или штор (из плотной ткани и одноцветных, гармонирующих с цветом стен), позволяющих закрывать оконные проемы
- располагать экраны в недоступной зоне для яркого света, во избежание перепада освещенности и бликов на экране
- расстояние от стен с оконными проемами не менее 1.5 м, от стен без оконных проемов – не менее 1.0 м
- освещенность экрана не менее 200 лк
- освещенность рабочего места, клавиатуры, стола не менее 400 лк

При необходимости допускается установка на рабочие места светильников с условием, что они не создают бликов на экране и увеличивают освещенность экрана до уровня не более 300 лк.

Отраслевые нормы рекомендуют освещенность в пределах 400-700 лк. Мощность используемых ламп до 40 Вт.

Пол в помещении должен быть ровным, без выбоин и иных дефектов с нескользкой поверхностью, обладающей антистатическими свойствами, легко чистящимся.

Организация рабочего места с ЭВМ:

- расстояние между мониторами не менее 2 м;
- расстояние между пользователем и экраном монитора не менее 0.5 *м*, рекомендуется 0.6-0.7 *м*;
- расстояние клавиатуры от края поверхности стола 10-30 см;
- рабочий стул/кресло должно иметь возможность регулирования по высоте, углу наклона спинки и сиденья, иметь возможность поворота;

- дисплей должен иметь возможность поворачиваться по горизонтали и вертикали в пределах 30˚, не иметь блестящих деталей, поверхность дисплея должна быть матовой и одного цвета.

Продолжительность непрерывной работы за компьютером устанавливается в зависимости от рода деятельности человека.

Продолжительность непрерывной работы с ПК для пользователя не должна превышать 2 часов. Следует менять время от времени позу и делать перерывы каждые 2 часа работы.

При продолжительности смены в 12 часов работы с ПК устанавливаются на каждый час перерыв в 15 мин в течение последних четырех часов. При работе с ПК в ночную смену продолжительность регламентированных перерывов увеличивается на 60 минут.

При возникновении дискомфорта, напряжения зрительного органа и других неприятных и болевых ощущениях следует применить индивидуальный подход в ограничении рабочего времени с ПК, при возможности изменить условия труда в лучшую сторону, изменить график и/или увеличить длительность перерывов. Данные меры возможны при соблюдении санитарно-гигиенических и эргономических требований к труду с использованием ПК.

ЗАКЛЮЧЕНИЕ

В современных условиях автоматизация административного управления в образовательном учреждении – объективная необходимость и является средством, с помощью которого обеспечивается выполнение функций управления: планирование, составление расписаний и отчетности, оперативное и диспетчерское управление и т.п. Эти функции объективно необходимы для обеспечения качественного учебного процесса и нормальных условий работы персонала.

Одним из основных показателей эффективности работы учреждения является его производительность: качество, количество и скорость обработки информации. Любая организация обрабатывает информацию для выработки двух видов «продукции»: информации (данных, документов, речевой информации) и решений (оперативных и стратегических). Организация получает начальную (входящую) информацию в различных видах: документы, содержащие информацию в виде слов и цифр; речевая информация с телефона; данные от ЭВМ в электронной форме. Конечная (исходная) информация получается в таких же видах.

В процессе дипломного проектирования были получены следующие результаты.

Проанализированы сущность задач документооборота в образовательном учреждении и исследована организация документооборота на примере государственного бюджетного общеобразовательного учреждения города Москвы "Школа № 2095 "Покровский квартал". Охарактеризовано входную, постоянную, промежуточную и исходную информацию и построено инфологическую модель системы.

На основе проведенного исследования разработано удобную в использовании и одновременно достаточно функциональную информационную систему "Документооборот" для операционной системы Windows®, которая

автоматизирует учет документооборота в заведении и позволяет осуществлять выборку и печать отдельных документов.

Программа разработана на языке Object Pascal в интегрированной среде Delphi с использованием возможностей языка SQL, не нуждается в мощной аппаратной поддержке. В ней применяется простой алгоритм поиска и фильтрации информации, присущий SQL запросам.

Планируется продолжение работы над программой, а именно разработка подсистем, автоматизирующих документооборот в других подразделениях учебного заведения.

В процессе разработки получены выводы, представлены далее:

Для каждой БД нужно отдельно подбирать СУБД, и способ организации БД так как в обширной сфере деятельности невозможно предусмотреть все варианты и сделать что-то идеальное, каждая СУБД имеет свои преимущества и недостатки.

Продуманная структура и правильно выбранный формат записей способствует эффективной работе с БД и позволяют быстро получать доступ к необходимой информации.

Перед разработкой БД с множеством таблиц, необходимо:

– определить цель создания БД;

– выбрать принцип группирования данных;

– продумать структуру таблиц;

– назначить ключевые поля в таблицах;

– пределить схему связей между таблицами.

В разработанной СУБД соблюдены все эти требования.

Программа пригодна к практической организации документооборота в образовательном учреждении, не требует значительных затрат на внедрение и обучение персонала.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абасова, С.Э. Информационные технологии в открытом образовании и дистанционном обучении. / С.Э. Абасова Бюллетень «Высшая школа»2006.- №1.-С. 127-128.

2. А.Я. Архангельский Приемы программирования в Delphi на основе VCL. – М. ООО «Бином-Пресс», 2006

3. Генкин Б. М. Г27 Экономика и социология труда : учеб. для вузов / Б. М. Генкин. — 7-е изд., доп. — М. : Норма, 2007. — 448 с.

4. Глушаков С. В. Delphi 2007 г. Москва: издательство «Хранитель», 2008 г. – 635 с.;

5. Епашников А., Епашнико В. Delphi. Программирование СУБД. – М.: Диалог-МИФИ, 2001. – 528 с.

6. Митчелл К. Керман Программирование и отладка в Delphi: Учебный курс: М.; СПб.; Киев, 2003.

7. Кузнецов В. В., Абдрашитова И.В. "Основы объектно-ориентированного программирования в DELPHI", "Школьный Университет, Томск, 2008. 461 с.

8. Фаронов В.В., Шумаков П.В. “Delphi 5. Руководство разработчика баз данных”, М.:“Ноллидж” – 2000, 635 с.

9. Фаронов В.В. Delphi. Программирование на языке высокого уровня: [Учеб. для вузов. Допущено МО РФ].- М. [и др.]: Питер, 2007.- 639с.

10. Фленов М.Е. Библия Delphi, 2-е изд., перераб. и доп. – СПб: БХВ – Петербург, 2008

11. Рейнгольд Л. Обзор систем электронного документооборота // iXBT. com [Электронный ресурс]. Режим доступа: ixbt. com/soft/sed. shtml.

12. Компьютерные истоки Старой площади // Альманах "Восток". Коммерсант. 2005. N 1/2 (25/26) [Электронный ресурс]. Режим доступа: situation. ru.

13. Назаренко А. История и тенденции развития современных СЭД // Интернет-издание IT Practice [Электронный ресурс]. Режим доступа: itpractice. ru.

14. Ричардсон Роберт. Документооборот для всех остальных // Журнал сетевых решений/LAN. 1998. N 2. [Электронный ресурс]. Режим доступа: osp. ru.

15. Мейнцер А. Корпоративные хранилища документов // Современные технологии делопроизводства и документооборота. 2010. N 0. С. 30 - 37.

16. Глоссарий терминов по хранению документации, Организация Объединенных Наций (ООН) [Электронный ресурс]. Режим доступа: un. org.

17. Храмцовская Н. А. Закон о государственных услугах и развитие делопроизводства // Секретарь-референт. 2010. N 10. С. 32 - 33.

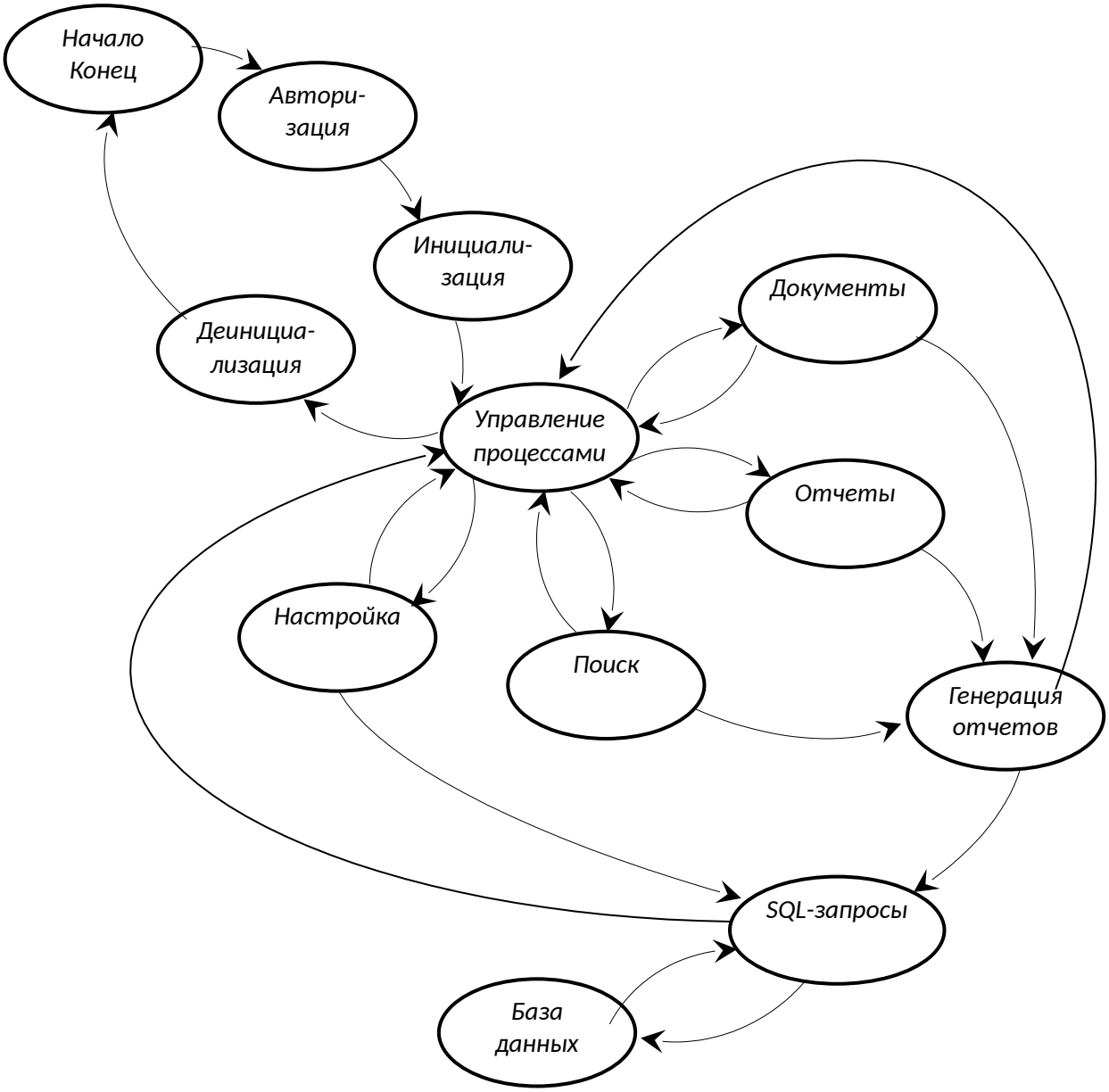
18. ГОСТ 2.105-95 ЕСКД. Общие требования к текстовым документам.

19. ГОСТ 7.1-84 Библиографическое описание документа. Общие требования и правила составления.

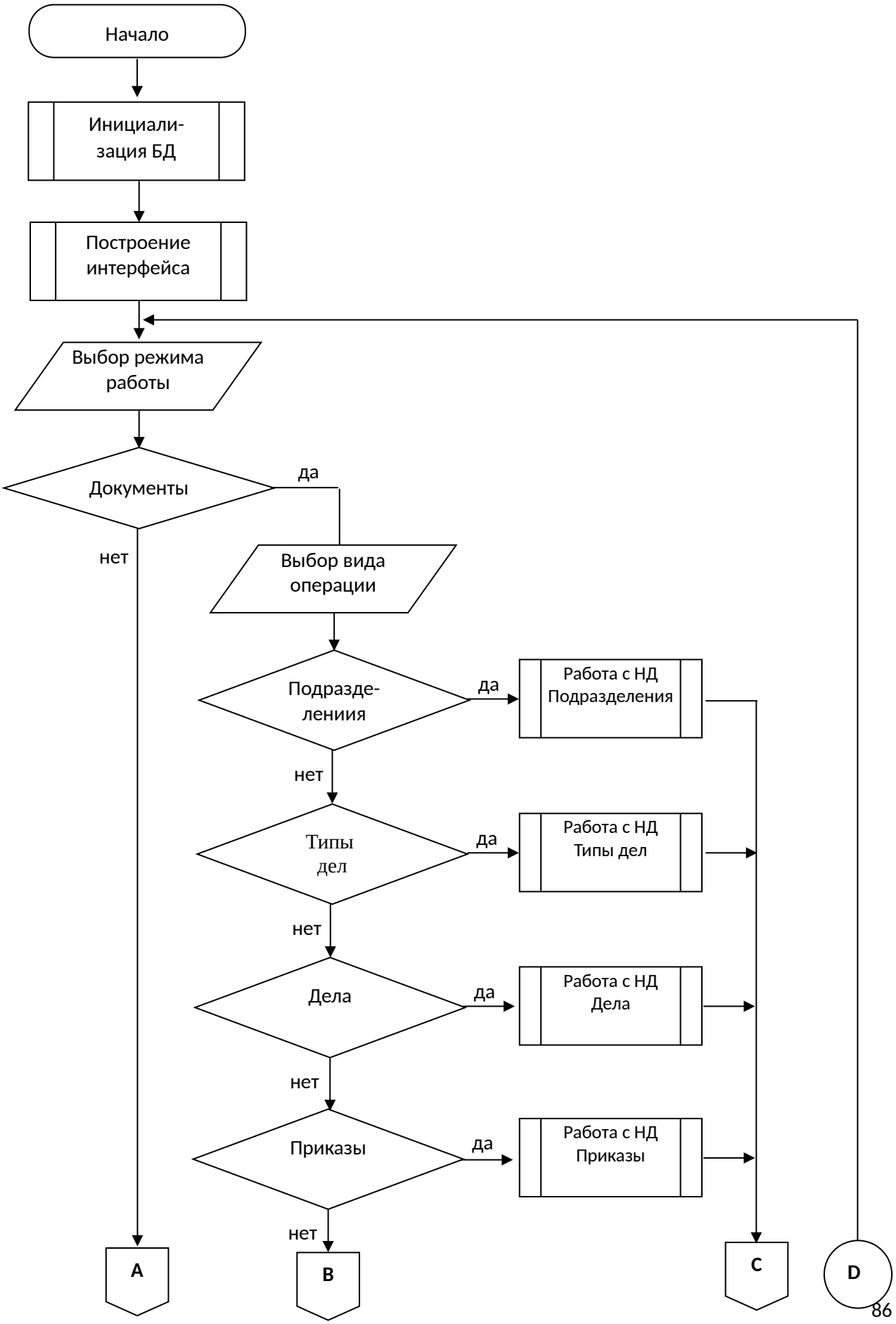
20. ГОСТ 7.32-91 Отчет о научно-исследовательской работе. Структура и правила оформления.

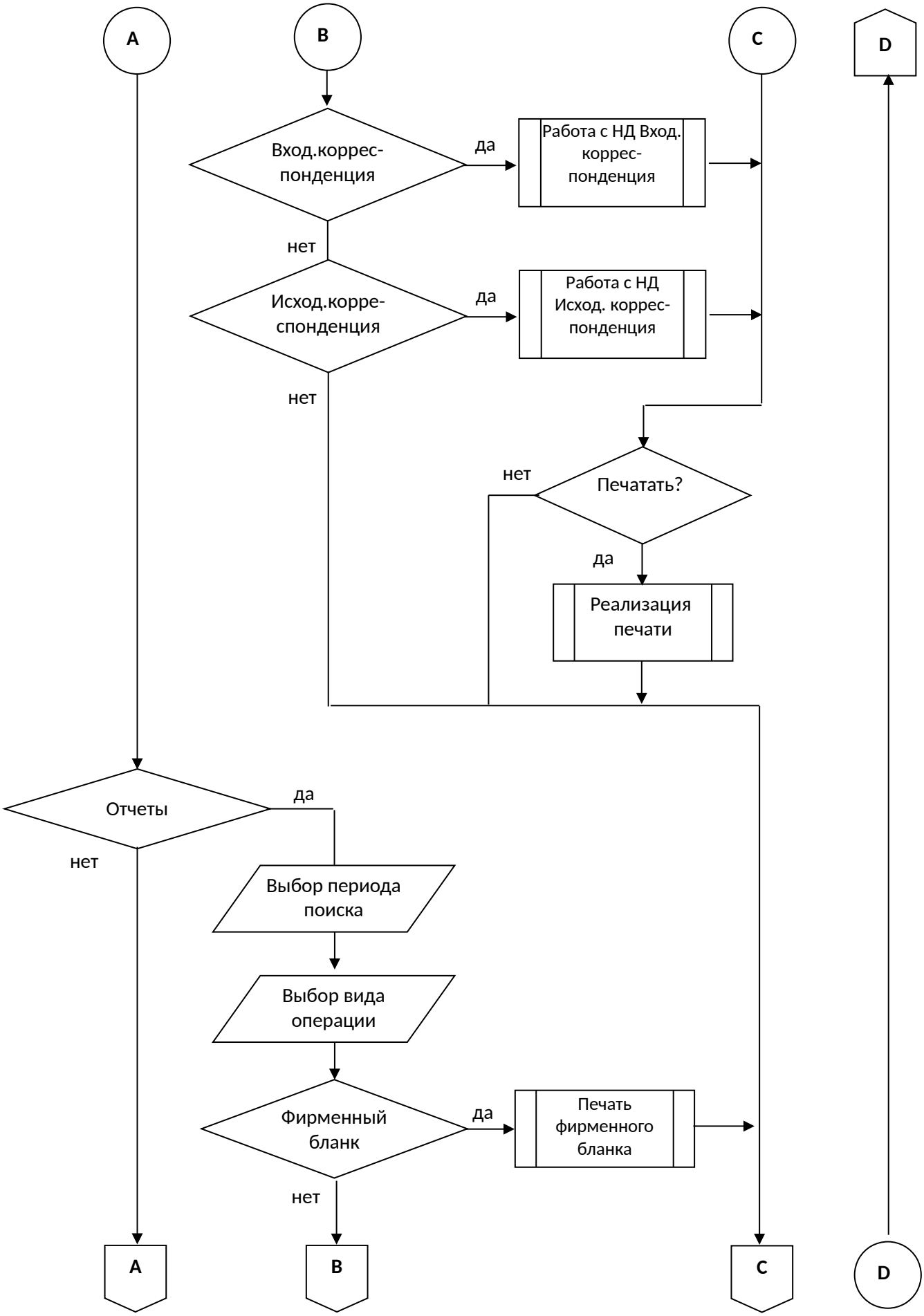
ПРИЛОЖЕНИЯ

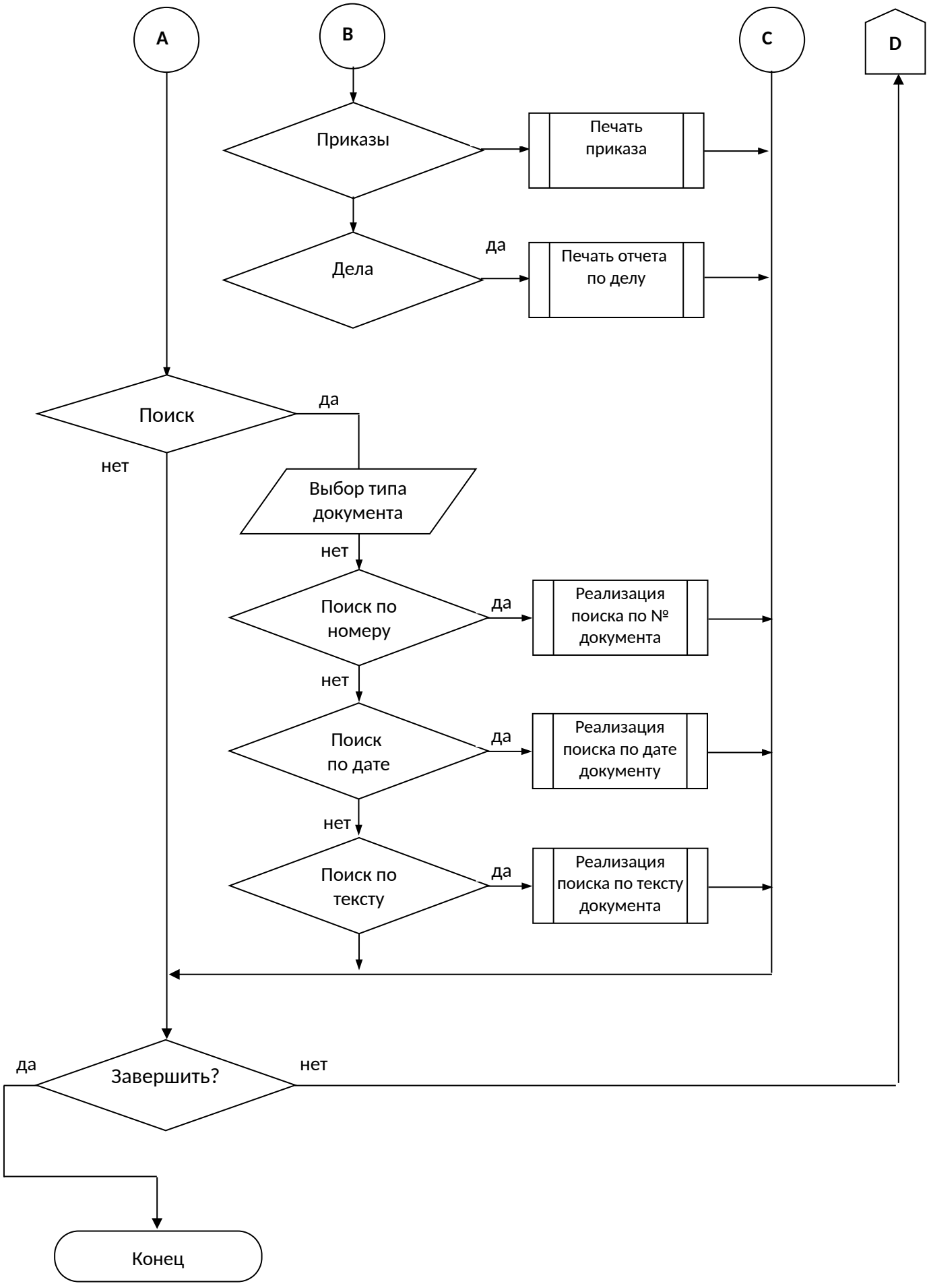
Приложение А – Диаграма процессов программы



Приложение Б – Блок-схема программы




A
B
C
D
Вход.коррес-понденция
да
Работа с НД Вход. коррес-понденция
нет
Исход.корре-спонденция
да
Работа с НД Исход. коррес-понденция
нет
нет
Печатать?
да
Реализация печати
Отчеты
да
нет
Выбор периода поиска
Выбор вида операции
Фирменный бланк
да
Печать фирменного бланка
нет
A
B
C
D


A
B
C
D
Приказы
Печать приказа
Дела
да
Печать отчета по делу
Поиск
да
нет
Выбор типа документа
нет
Поиск по номеру
да
Реализация поиска по № документа
нет
Поиск по дате
да
Реализация поиска по дате документу
нет
Поиск по тексту
да
Реализация поиска по тексту документа
да
Завершить?
нет
Конец

Приложение В – Исходный код программы

```
unit UMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, IBDatabase, DB, IBCustomDataSet, IBQuery, ComCtrls,
  ExtCtrls, StdCtrls, Grids, Buttons, Mask, IniFiles,math, Word2000,
  OleServer, Spin, IdBaseComponent, IdComponent, IdTCPConnection,
  IdTCPClient, IdMessageClient, IdSMTP,IdMessage, IdPOP3, DBGrids, DBCtrls;

type
  TFMain = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Open1: TMenuItem;
    Settings1: TMenuItem;
    Archiv1: TMenuItem;
    N1: TMenuItem;
    Close1: TMenuItem;
    Docs1: TMenuItem;
    Report1: TMenuItem;
    Help1: TMenuItem;
    Content1: TMenuItem;
    About1: TMenuItem;
    StatusBar1: TStatusBar;
    Find1: TMenuItem;
    PC1: TPageControl;
    TSDocs: TTabSheet;
    PC3: TPageControl;
    TS31: TTabSheet;
    TS32: TTabSheet;
    TS33: TTabSheet;
    TS34: TTabSheet;
    TS35: TTabSheet;
    TS36: TTabSheet;
    TSReport: TTabSheet;
    TSFind: TTabSheet;
    Panel1: TPanel;
    Button1: TButton;
    RB1: TRadioButton;
    E_FindNum: TEdit;
    RB2: TRadioButton;
    E_FindText: TEdit;
    RB3: TRadioButton;
    SGFind: TStringGrid;
    Panel2: TPanel;
    SB1: TSpeedButton;
    SB2: TSpeedButton;
    SB3: TSpeedButton;
    SB4: TSpeedButton;
    SB5: TSpeedButton;
    SB6: TSpeedButton;
    SB7: TSpeedButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    ME21: TMaskEdit;
```

```
ME22: TMaskEdit;
TS7: TTabSheet;
SaveDialog1: TSaveDialog;
Label1: TLabel;
Label6: TLabel;
CB1: TComboBox;
Label7: TLabel;
Panel3: TPanel;
Label12: TLabel;
Label10: TLabel;
E_DepName: TEdit;
Label11: TLabel;
M_DepInfo: TMemo;
Panel4: TPanel;
Panel5: TPanel;
Label13: TLabel;
Panel6: TPanel;
Label14: TLabel;
Label15: TLabel;
Label16: TLabel;
Label17: TLabel;
Label18: TLabel;
Label19: TLabel;
Label20: TLabel;
Label21: TLabel;
E_TypeCase_Index: TEdit;
B_AddDep: TButton;
Button8: TButton;
M_TypeCaseName: TMemo;
E_TypeCase_Period: TEdit;
E_TypeCase_Num: TEdit;
E_Info: TEdit;
B_AddTypeCase: TButton;
Panel7: TPanel;
Label22: TLabel;
B_RefreshTypeCases: TButton;
Panel8: TPanel;
Panel9: TPanel;
Label23: TLabel;
Label24: TLabel;
Label27: TLabel;
Label29: TLabel;
Label30: TLabel;
Label31: TLabel;
Label32: TLabel;
Label33: TLabel;
E_NumCase: TEdit;
CB_IndexCase: TComboBox;
M_InfoCase: TMemo;
ME_DateCase: TMaskEdit;
Label34: TLabel;
Label35: TLabel;
Label36: TLabel;
Label37: TLabel;
Label38: TLabel;
Label39: TLabel;
Label40: TLabel;
ME_InCorDate1: TMaskEdit;
E_Sender: TEdit;
E_InCorFhN: TEdit;
ME_InCorDate2: TMaskEdit;
M_InCorText: TMemo;
```

```
E_InCorSign: TEdit;
Panel12: TPanel;
Label42: TLabel;
Label43: TLabel;
Label44: TLabel;
Label45: TLabel;
ME_OutCorDate: TMaskEdit;
E_SendTo: TEdit;
M_OutCorText: TMemo;
SGZ: TStringGrid;
Panel14: TPanel;
Label46: TLabel;
E_NumDecree: TEdit;
Edit5: TEdit;
WordDocument1: TWordDocument;
WordParagraphFormat1: TWordParagraphFormat;
WordApplication1: TWordApplication;
Label47: TLabel;
Label48: TLabel;
ME_FindDate: TMaskEdit;
SE_TypeCase_CountPart: TSpinEdit;
Timer1: TTimer;
Button18: TButton;
IdSMTP1: TIdSMTP;
Button2: TButton;
Mail1: TMenuItem;
Connect1: TMenuItem;
Disconnect1: TMenuItem;
Button25: TButton;
Msg: TIdMessage;
DBG_Dep: TDBGrid;
Button6: TButton;
Button3: TButton;
B_EditTypeCase: TButton;
B_DelTypeCase: TButton;
DBNavigator1: TDBNavigator;
DBG_TypeCases: TDBGrid;
DBLkpCB_Dep: TDBLookupComboBox;
B_PrintTypeCases: TButton;
DBNavigator2: TDBNavigator;
Button5: TButton;
Panel15: TPanel;
Label5: TLabel;
DBNavigator3: TDBNavigator;
DBG_Decrees: TDBGrid;
Label26: TLabel;
ME_DateDecree: TMaskEdit;
B_NewCase: TButton;
B_PrintCase: TButton;
B_EditCase: TButton;
Panel10: TPanel;
Label8: TLabel;
DBNavigator4: TDBNavigator;
DBG_Cases: TDBGrid;
M_DecreeText: TMemo;
Button20: TButton;
B_SaveDecree: TButton;
B_PrintSaveDecree: TButton;
DBLkpCB_Dep2: TDBLookupComboBox;
Panel11: TPanel;
Label9: TLabel;
DBNavigator5: TDBNavigator;
```

```
DBG_InCors: TDBGrid;
B_NewInCor: TButton;
B_EditInCor: TButton;
DBLkpCB_Executors: TDBLookupComboBox;
B_DelInCor: TButton;
Panel13: TPanel;
Label25: TLabel;
DBNavigator6: TDBNavigator;
DBG_OutCors: TDBGrid;
DBLkpCB_Dep3: TDBLookupComboBox;
B_NewOutCor: TButton;
B_EditOutCor: TButton;
B_DelOutCor: TButton;
B_EditDecree: TButton;
B_DelDecree: TButton;
B_DelCase: TButton;
DBLkpCB_Dep4: TDBLookupComboBox;
procedure Close1Click(Sender: TObject);
procedure Open1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Settings1Click(Sender: TObject);
procedure Archiv1Click(Sender: TObject);
procedure Content1Click(Sender: TObject);
procedure About1Click(Sender: TObject);
procedure Report1Click(Sender: TObject);
procedure Find1Click(Sender: TObject);
procedure Docs1Click(Sender: TObject);
procedure Panel8Resize(Sender: TObject);
procedure B_AddDepClick(Sender: TObject);
{}FUNCTION MAKEDATE(S:STRING):STRING;
{}FUNCTION UNMAKEDATE(S:STRING):STRING;
{}procedure SGZClear;
procedure Button6Click(Sender: TObject);
procedure Button8Click(Sender: TObject);
procedure Panel3Resize(Sender: TObject);
procedure B_AddTypeCaseClick(Sender: TObject);
procedure B_RefreshTypeCasesClick(Sender: TObject);
procedure B_EditTypeCaseClick(Sender: TObject);
procedure B_SaveDecreeClick(Sender: TObject);
procedure B_PrintSaveDecreeClick(Sender: TObject);
procedure B_NewCaseClick(Sender: TObject);
procedure B_NewInCorClick(Sender: TObject);
procedure SB1Click(Sender: TObject);
procedure SB2Click(Sender: TObject);
procedure SB3Click(Sender: TObject);
procedure SB4Click(Sender: TObject);
procedure SB5Click(Sender: TObject);
procedure SB6Click(Sender: TObject);
procedure SB7Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
{}procedure Word1( s:string;a, b:OleVariant;var vstart, vend: OleVariant);
procedure Timer1Timer(Sender: TObject);
{}procedure WordInsertText(Text: string);
{}procedure WordGotoBookmark(Bookmark: string);
{}procedure WordInit(FileName:OleVariant);
{}procedure WordEnd(FileName:OleVariant);
{}procedure TableF(U:integer);
procedure B_PrintTypeCasesClick(Sender: TObject);
procedure B_PrintCaseClick(Sender: TObject);
procedure Button18Click(Sender: TObject);
```

```
    {}procedure print2;
    {}procedure print3;
    {}procedure print4;
    {}procedure print5;
    {}procedure print6;
    {}procedure print7;
    procedure E_FindNumClick(Sender: TObject);
    procedure E_FindTextClick(Sender: TObject);
    procedure ME_FindDateClick(Sender: TObject);
    procedure Button20Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure B_DelTypeCaseClick(Sender: TObject);
    procedure Connect1Click(Sender: TObject);
    procedure IdSMTP1Connected(Sender: TObject);
    procedure IdSMTP1Disconnected(Sender: TObject);
    procedure Button25Click(Sender: TObject);
    procedure B_EditCaseClick(Sender: TObject);
    procedure Disconnect1Click(Sender: TObject);
    procedure IdSMTP1WorkEnd(Sender: TObject; AWorkMode: TWorkMode);
    {}function decoder(s:string):string;
    {}function coder(s:string):string;
    procedure DBG_DepDblClick(Sender: TObject);
    procedure DBG_TypeCasesDblClick(Sender: TObject);
    procedure DBG_DecreesDblClick(Sender: TObject);
    procedure DBG_CasesDblClick(Sender: TObject);
    procedure ME_DateDecreeChange(Sender: TObject);
    procedure DBLkpCB_Dep2Exit(Sender: TObject);
    procedure B_EditInCorClick(Sender: TObject);
    procedure B_NewOutCorClick(Sender: TObject);
    procedure DBG_InCorsDblClick(Sender: TObject);
    procedure B_EditOutCorClick(Sender: TObject);
    procedure B_EditDecreeClick(Sender: TObject);
    procedure B_DelDecreeClick(Sender: TObject);
    procedure B_DelCaseClick(Sender: TObject);
    procedure B_DelInCorClick(Sender: TObject);
    procedure B_DelOutCorClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    tn: Array[1..1000] of integer;
    tn2,tn3:Array[0..100] of string;
    Dir:string;
    PathDB:string;
    pwd, DirectorName, TelBookPath, Delovod,
    SMTPSer, Login, Pass, Addr:string;
    A:integer;
  end;

var
  FMain: TFMain;
  PathProg:string;

implementation

uses USettings, UContent, UAbout, UExecutor, UAccess, UData;

{$R *.dfm}
```

```
{================MY PROCEDURES=================}

function TFMain.decoder(s:string):string;
  var i:integer; p:string;
begin
  p:='';
  for i:=1 to length(s) do
    p:=p+chr(ord(s[i])+i);
  decoder:=p;
end;

function TFMain.coder(s:string):string;
  var i:integer; p:string;
begin
  p:='';
  for i:=1 to length(s) do
  p:=p+chr(ord(s[i])-i);
  coder:=p;
end;

procedure TFMain.SGZClear;
  var i:integer;
begin
  SGZ.Cells[0,1]:='';
  SGZ.Cells[1,1]:='';
  SGZ.Cells[2,1]:='';
  SGZ.Cells[3,1]:='';
  SGZ.Cells[4,1]:='';
  for i:=0 to SGZ.Colcount-1 do
    SGZ.ColWidths[i]:=100;
end;

procedure TFMain.Word1( s:string;a, b:OleVariant;var vstart, vend: OleVariant);
  var j, ilengy: Integer;
begin
  ilengy := Length(WordDocument1.Range.Text);
  for j := 0 to ilengy - 8 do
  begin
    a := j;
    b := j + length(s);
    if WordDocument1.Range(a, b).Text = s then
    begin
      vstart := j;
      vend := j + 7;
    end;
  end;
end;

procedure TFMain.WordInsertText(Text: string);
begin
  WordApplication1.Selection.TypeText(Text);
end;

procedure TFMain.WordGotoBookmark(Bookmark: string);
  var What : OLEVariant;
      Which : OLEVariant;
      Count : OLEVariant;
      Name : OLEVariant;
begin
  What := wdGoToBookmark;
  Which := unAssigned;
  Count := unAssigned;
```

```
  Name := Bookmark;
  WordApplication1.Selection.GoTo_(What, Which, Count, Name);
end;

procedure TFMain.WordInit(FileName:OleVariant);
begin
 {Èíèöèàëèçàöèÿ}
  WordApplication1.Connect;
  WordApplication1.Documents.Open(FileName, EmptyParam,
  EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam,
  EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam);

  WordDocument1.ConnectTo(WordApplication1.ActiveDocument);
  {Îòêëþ÷àåì ïðîâåðêó îðôîãðàôèè}
  WordApplication1.Options.CheckSpellingAsYouType:=False;
  WordApplication1.Options.CheckGrammarAsYouType:=False;
  {END Èíèöèàëèçàöèÿ}
end;

procedure  TFMain.WordEnd(FileName:OleVariant);
begin
 WordDocument1.SaveAs(FileName);
 WordApplication1.Options.CheckSpellingAsYouType:=True;
 WordApplication1.Options.CheckGrammarAsYouType:=True;
 Timer1.Enabled:=True;  //Äëÿ òîãî, ÷òîáû íå ïîÿâëÿëîñü îêíî Âîðäà
end;


procedure TFMain.TableF(U:integer);
var tformat, tappbrd: OleVariant;
begin
  tformat := wdTableFormatGrid1;
  tappbrd := wdTableFormatApplyBorders;

  WordDocument1.Tables.Item(U).AutoFormat(tformat, tappbrd, EmptyParam,
  EmptyParam, EmptyParam, EmptyParam,
  EmptyParam, EmptyParam, EmptyParam, EmptyParam); //format of table
end;

{====================END====================}
procedure TFMain.Close1Click(Sender: TObject);
begin
 close;
end;

procedure TFMain.Open1Click(Sender: TObject);
var
 IniFile: Tinifile;
begin
 FAccess.Showmodal;

 if A=1
 then begin
      IniFile:=TiniFile.Create(extractfilepath(Application.ExeName)+
'zauch.ini');
      PathDB:=IniFile.ReadString('Location', 'PathDB', 'zauch.gdb');
      IniFile.Free;

      with DM do
      begin
        IBDB.DatabaseName:=PathDB;
        IBDB.Open;
```

```
        IBQ_Dep.Open;
          IBQ_Executors.Open;

        IBQ_TypeCases.Open;
        IBQ_Decrees.Open;
        IBQ_Cases.Open;
        IBQ_InCors.Open;
      end;
      end;
end;

procedure TFMain.FormCreate(Sender: TObject);
var
 IniFile: Tinifile; // ñîçäàåì ïåðåìåííóþ òèïà Tinifile
 //InputString:string;
begin
 PathProg:=GetCurrentDir;

 IniFile:=TiniFile.Create(extractfilepath(Application.ExeName) + 'zauch.ini');
 FMain.Left    :=IniFile.ReadInteger('Position', 'X', 50);
 FMain.Top     :=IniFile.ReadInteger('Position', 'Y', 50);
 DirectorName  := IniFile.ReadString('Director', 'Name', '_____________');
 Delovod       := IniFile.ReadString('Director', 'Delovod', '');
 SMTPSer       := IniFile.ReadString('E-mail', 'SMTP', '');
 Login         := IniFile.ReadString('E-mail', 'Login', '');
 Addr          := IniFile.ReadString('E-mail', 'Address', '');
 Pass          := decoder(IniFile.ReadString('E-mail', 'Password', ''));
 pwd           := decoder(IniFile.ReadString('Program', 'PWD', ''));
 IniFile.Free;

 GetDir(0,Dir);
 SaveDialog1.InitialDir := extractfilepath(Application.ExeName) + 'Archiv';

 Docs1.Enabled   := False;
 Report1.Enabled := False;
 Find1.Enabled := False;
 PC1.Visible     := False;

 {Title of StringGrid for Department}
 PC3.ActivePageIndex :=6;
end;

procedure TFMain.FormClose(Sender: TObject; var Action: TCloseAction);
VAR
IniFile: Tinifile;
begin
 IniFile:=TiniFile.Create(extractfilepath(Application.ExeName) + 'zauch.ini');
 IniFile.WriteInteger('Position', 'X', FMain.Left);
 IniFile.WriteInteger('Position', 'Y', FMain.Top);

 IniFile.Free;
 DM.IBDB.Connected := False;
 IdSMTP1.Disconnect;
end;

procedure TFMain.Settings1Click(Sender: TObject);
begin
 FSettings.ShowModal;
end;

procedure TFMain.Archiv1Click(Sender: TObject);
var FileS, FileArch: string;
```

```
begin
 FileS    := PathDB;
 if SaveDialog1.Execute then
  begin
    FileArch := SaveDialog1.FileName;
  end;

  if CopyFile(pchar(FileS),pchar(FileArch),true)
  then ShowMessage('Àðõèâîóþ êîïèþ ñîçäàíî!')
  else ShowMessage('Îøèáêà ïðè ñîçäàíèè àðõèâíîé êîïèè!');
end;

procedure TFMain.Content1Click(Sender: TObject);
begin
 FContent.SHowModal;
end;

procedure TFMain.About1Click(Sender: TObject);
begin
 FAbout.ShowModal;
end;

procedure TFMain.Report1Click(Sender: TObject);
begin
 PC1.ActivePageIndex :=1;
end;

procedure TFMain.Find1Click(Sender: TObject);
begin
 PC1.ActivePageIndex :=2;
end;

procedure TFMain.Docs1Click(Sender: TObject);
begin
 PC1.ActivePageIndex :=0;
end;

procedure TFMain.Panel8Resize(Sender: TObject);
begin
end;

procedure TFMain.B_AddDepClick(Sender: TObject);
begin
  if E_DepName.Text=''
  then exit;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
       SQL.Add(format('INSERT INTO DEPARTS (DEP,INFO) VALUES(''%s'', ''%s'')',
[E_DepName.Text, M_DepInfo.Text]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Dep.Close;
    IBQ_Dep.Open;
  end;
```

```
  E_DepName.Text  :='';
  M_DepInfo.Text  :='';
end;

procedure TFMain.Button6Click(Sender: TObject);
  var s:string;
begin
  if DM.IBQ_Dep.IsEmpty
  then exit;

  if E_DepName.Text=''
  then exit;

  with DM, DM.IBQuery1 do
  begin
       s:='UPDATE  DEPARTS  SET  DEP='''  +  E_DepName.Text  +  ''',  INFO='''  +
M_DepInfo.Text + ''' WHERE ID=' + DM.IBQ_Dep.FieldByName('ID').AsString;
    SQL.Clear;
    SQL.Add(s);
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Dep.Close;
    IBQ_Dep.Open;
  end;
end;

procedure TFMain.Button3Click(Sender: TObject);
begin
  if DM.IBQ_Dep.IsEmpty
  then exit;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
                  SQL.Add('DELETE    FROM    DEPART    WHERE    NUMBER='    +
DM.IBQ_Dep.FieldByName('ID').AsString);
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Dep.Close;
    IBQ_Dep.Open;
  end;

  E_DepName.Text  :='';
  M_DepInfo.Text  :='';
end;


procedure TFMain.Button8Click(Sender: TObject);
begin
```

```
   DM.IBQ_Dep.Close;
   DM.IBQ_Dep.Open;
end;

procedure TFMain.Panel3Resize(Sender: TObject);
begin
end;

procedure TFMain.B_AddTypeCaseClick(Sender: TObject);
begin
        if    (M_TypeCaseName.Text='')    or    (E_TypeCase_Index.Text='')    or
(E_TypeCase_Period.Text='')
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
                                                  SQL.Add('INSERT            INTO
TYPECASES(DEP,IND,COUNTPART,NAMECASE,PERIOD,PERNUM,INFO)');
    SQL.Add(format('VALUES(%s, ''%s'', %s, ''%s'', ''%s'', ''%s'', ''%s'')',
               [IntToStr(DBLkpCB_Dep.KeyValue),
                E_TypeCase_Index.Text,
                SE_TypeCase_CountPart.Text,
                M_TypeCaseName.Text,
                E_TypeCase_Period.Text,
                E_TypeCase_Num.Text,
                E_Info.Text
               ]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_TypeCases.Close;
    IBQ_TypeCases.Open;
  end;

  DBLkpCB_Dep.KeyValue:=-1;
  E_TypeCase_Index.Text:='';
  SE_TypeCase_CountPart.Value:=1;
  M_TypeCaseName.Text:='';
  E_TypeCase_Period.Text:='';
  E_TypeCase_Num.Text:='';
  E_Info.Text:='';
end;

procedure TFMain.B_RefreshTypeCasesClick(Sender: TObject);
// Ïîêàçûâàòü âñå òèïû äåë
begin
  DM.IBQ_TypeCases.Close;
  DM.IBQ_TypeCases.Open;
end;

procedure TFMain.B_EditTypeCaseClick(Sender: TObject);
begin
  if DM.IBQ_TypeCases.IsEmpty
```

```
  then exit;

        if    (M_TypeCaseName.Text='')    or    (E_TypeCase_Index.Text='')    or
(E_TypeCase_Period.Text='')
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
       SQL.Add(format('Update  TYPECASES  Set  DEP=%s,  IND=''%s'',  COUNTPART=%s,
NAMECASE=''%s'', PERIOD=''%s'', PERNUM=''%s'', INFO=''%s'' Where ID=%s',
              [
                IntToStr(DBLkpCB_Dep.KeyValue),
                E_TypeCase_Index.Text,
                SE_TypeCase_CountPart.Text,
                M_TypeCaseName.Text,
                E_TypeCase_Period.Text,
                E_TypeCase_Num.Text,
                E_Info.Text,
                IBQ_TypeCasesID.AsString
              ]));
    try  //ShowMessage(SQL.Text);
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_TypeCases.Close;
    IBQ_TypeCases.Open;
  end;
end;

procedure TFMain.B_DelTypeCaseClick(Sender: TObject);
begin
  with DM, DM.IBQuery1 do
  begin
    if IBQ_TypeCases.IsEmpty
    then exit;

    SQL.Clear;
    SQL.Add('DELETE FROM TYPECASES WHERE ID=' + DM.IBQ_TypeCasesID.AsString);
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_TypeCases.Close;
    IBQ_TypeCases.Open;

    DBLkpCB_Dep.KeyValue:=-1;
    E_TypeCase_Index.Text:='';
    SE_TypeCase_CountPart.Value:=1;
    M_TypeCaseName.Text:='';
    E_TypeCase_Period.Text:='';
```

```
    E_TypeCase_Num.Text:='';
    E_Info.Text:='';
  end;
end;

procedure TFMain.B_SaveDecreeClick(Sender: TObject);
begin
  if (E_NumDecree.Text = '') or (M_DecreeText.Text='')
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
    SQL.Add('INSERT INTO DECREES(DECREE_NUM, DECREE_TEXT, DATE_)');
    SQL.Add(format('VALUES(''%s'', ''%s'', ''%s'')',
               [E_NumDecree.Text,
                M_DecreeText.Text,
                ME_DateDecree.Text
               ]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Decrees.Close;
    IBQ_Decrees.Open;
  end;

  E_NumDecree.Text:='';
  M_DecreeText.Text:='';
  ME_DateDecree.Text:='';
end;


procedure TFMain.B_NewCaseClick(Sender: TObject);
begin
  if (DBLkpCB_Dep2.KeyValue=-1) or (E_NumCase.Text ='') or (ME_DateCase.Text='')
or
     (CB_IndexCase.Text='') or (M_InfoCase.Text='')
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
    SQL.Add('INSERT INTO CASES(DEP, CASE_NUM, IND, CASE_DATE, CASE_TEXT)');
    SQL.Add(format('VALUES(%s, ''%s'', ''%s'', ''%s'', ''%s'')',
               [IntToStr(DBLkpCB_Dep2.KeyValue),
                E_NumCase.Text,
                CB_IndexCase.Text,
                ME_DateCase.Text,
                M_InfoCase.Text
               ]));
    try
```

```
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Cases.Close;
    IBQ_Cases.Open;
  end;

  DBLkpCB_Dep2.KeyValue:=-1;
  E_NumCase.Text:='';
  CB_IndexCase.Text:='';
  ME_DateCase.Text:='';
  M_InfoCase.Text:='';
end;

procedure TFMain.B_NewInCorClick(Sender: TObject);
begin
  if (ME_InCorDate1.Text='') or (E_Sender.Text='') or
     (E_InCorFhN.Text='') or (ME_InCorDate2.Text='') or
     (E_InCorSign.Text='') or (M_InCorText.Text='') or
     (DBLkpCB_Executors.KeyValue=-1)
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
       SQL.Add('INSERT  INTO  INCORS(DATE1,  SENDER,  NUMVH,  DATE2,  INCOR_TEXT,
ID_EXECUTOR, INFO)');
    SQL.Add(format('VALUES(''%s'', ''%s'', ''%s'', ''%s'', ''%s'', %s, ''%s'')',
               [ME_InCorDate1.Text,
                E_Sender.Text,
                E_InCorFhN.Text,
                ME_InCorDate2.Text,
                M_InCorText.Text,
                IntToStr(DBLkpCB_Executors.KeyValue),
                ''
               ]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_InCors.Close;
    IBQ_InCors.Open;
  end;

  ME_InCorDate1.Text:='';
  E_Sender.Text:='';
  E_InCorFhN.Text:='';
  ME_InCorDate2.Text:='';
  M_InCorText.Text:='';
  DBLkpCB_Executors.KeyValue:=-1;
end;
```

```
procedure TFMain.SB2Click(Sender: TObject);
//Ïðèêàçû çà ïåðèîä
  var s:string;
      i:integer; dt:TDate;
begin
  try
     dt:=StrToDate(ME21.Text);
     dt:=StrToDate(ME22.Text);
  except
    ShowMessage('Íåêîððåêòíàÿ äàòà!');
    exit;
  end;

  SGZ.RowCount:=2;
  SGZClear;

  SGZ.Tag :=2; //pokazatel chto budet print nakazy
  SGZ.ColCount := 4;
  SGZ.Cells[0,0]:='¹ ï/ï';
  SGZ.Cells[1,0]:='Íîìåð';
  SGZ.Cells[2,0]:='Äàòà';
  SGZ.Cells[3,0]:='Òåêñò(êðàòêî)';
  SGZ.ColWidths[3]:=400;

   s:='Select DECREE_NUM, DATE_, DECREE_TEXT from DECREES where (DATE_ between
''' + ME21.Text + ''' and ''' + ME22.Text + ''') order by DATE_';

  WITH DM.IBQuery1 DO
  BEGIN
    SQL.Clear;
    SQL.Add('COMMIT;');
    ExecSQL;

    SQL.Clear;
    SQL.Add(s);
    Open;

    First;
    i:=1;

    while not EOF do
    begin
      SGZ.RowCount:=i+1;
      SGZ.Cells[0,i] :=IntToStr(i);
      SGZ.Cells[1,i] :=Fields[0].AsString;
      SGZ.Cells[2,i] :=Fields[1].AsString;
      SGZ.Cells[3,i] :=Copy(Fields[2].AsString, 1, 128);
      inc(i);
      Next;
    end;

    SQL.Clear;
    SQL.Add('COMMIT;');
    ExecSQL;
  END;

end;

procedure TFMain.SB3Click(Sender: TObject);
  var s:string;
      tmp:array[1..200] of INTEGER;
      a,b,c:INTEGER;
```

```
      dt:TDate;
begin
  try
     dt:=StrToDate(ME21.Text);
     dt:=StrToDate(ME22.Text);
  except
    ShowMessage('Íåêîððåêòíàÿ äàòà!');
    exit;
  end;

  if DBLkpCB_Dep4.KeyValue<>-1 then
  begin
    SGZClear;
    SGZ.RowCount:=5;
    SGZ.Tag :=3; //pokazatel chto budet print srok hraneniya
    SGZ.ColCount := 2;
    SGZ.Cells[0,0]:='Ïî ñðîêàì õðàíåíèÿ';
    SGZ.Cells[1,0]:='Âñåãî';
    SGZ.Cells[0,1]:='Ïîñòîÿííîãî';
    SGZ.Cells[0,2]:='Äëèòåëüíîãî (ñâûøå 10 ëåò)';
    SGZ.Cells[0,3]:='Âðåìåííîãî (äî 10 ëåò âêëþ÷èòåëüíî)';
    SGZ.Cells[0,4]:='âìåñòå';
    SGZ.ColWidths[0]:=300;

             s:='select   IND,   PERIOD   from   TYPECASES   where   DEP='   +
IntToStr(DBLkpCB_Dep4.KeyValue);
    // Showmessage(s);
    with DM.IBQuery1 do
    begin
      SQL.Clear;   SQL.Add(s);   Open;
      First;
      while not EOF do
      begin
      if Fields[1].AsString='ïîñò'
      then TMP[Fields[0].AsInteger]:=200
      else TMP[Fields[0].AsInteger]:=Fields[1].asInteger;
      Next;
      end;
    end;

    A:=0; B:=0; C:=0;
     s:='select IND from CASES where (DEP=' + IntToStr(DBLkpCB_Dep4.KeyValue) +
') and (CASE_DATE between ''' + ME21.Text + ''' and ''' + ME22.Text + ''')';
    with DM.IBQuery1 do
    begin
      SQL.Clear;  SQL.Add(s);  Open;
      First;
      while not EOF do
      begin
      if TMP[Fields[0].AsINTEGER]<=10
      then inc(C)
      else if TMP[Fields[0].AsINTEGER]<=199
           then inc(B)
           else inc(A);
      Next;
      end;
      SGZ.Cells[1,1]:=IntToStr(A);
      SGZ.Cells[1,2]:=IntToStr(B);
      SGZ.Cells[1,3]:=IntToStr(C);
      SGZ.Cells[1,4]:=IntToStr(A + B + C);
    end;
```

```
  end
  else ShowMEssage('Âèáðèòå ïîäðàçäåë');
end;

procedure TFMain.SB4Click(Sender: TObject);
  var s,s1:WideString;
      i:integer;
begin
  SGZ.RowCount:=2;
  SGZClear;

  SGZ.Tag :=4; //pokazatel chto budet print cases
  SGZ.ColCount := 5;
  SGZ.Cells[0,0]:='¹ ï/ï';
  SGZ.Cells[1,0]:='Íîìåð';
  SGZ.Cells[2,0]:='Äàòà';
  SGZ.Cells[3,0]:='Èíäåêñ';
  SGZ.Cells[4,0]:='Òåêñò(êðàòêî)';
  SGZ.ColWidths[4]:=400;

  s1:='where CASE_DATE between ''' + ME21.Text + ''' and ''' + ME22.Text + '''';
    s  :='where  (DEP='  +  IntToStr(DBLkpCB_Dep4.KeyValue)  +  ')  and  (CASE_DATE
between ''' + ME21.Text + ''' and ''' + ME22.Text + ''')';
  if DBLkpCB_Dep4.KeyValue=-1 then s:=s1;

  with DM.IBQuery1 do
  begin
    //SQL.Clear; SQL.Add('COMMIT;'); ExecSQL;
    SQL.Clear;
    SQL.Add('select CASE_NUM, CASE_DATE, DEP, IND, CASE_TEXT from CASES');
    SQL.Add(s);
    SQL.Add('order by CASE_DATE');
    Open;

    First;
    i:=1;
    while not EOF do
    begin
      SGZ.RowCount:=i+1;
      SGZ.Cells[0,i] :=IntToStr(i);
      SGZ.Cells[1,i] :=Fields[0].AsString;
      SGZ.Cells[2,i] :=Fields[1].AsString;
      SGZ.Cells[3,i] :=Fields[2].AsString + '-' + Fields[3].AsString;;
      SGZ.Cells[4,i] :=COpy(Fields[4].AsString,1,128);
      inc(i);
      Next;
    end;
  end;
end;


procedure TFMain.SB5Click(Sender: TObject);
//Ïîñòïýíîãî õðàíåíèÿ
  var s:string;
      i:integer;
      TMP:array[1..100] of integer;
      tmp2:array[1..100,1..100] of integer;
begin
  SGZ.RowCount:=2;
  SGZClear;

  SGZ.Tag :=5; //pokazatel chto budet print case
```

```
  SGZ.ColCount := 5;
  SGZ.Cells[0,0]:='¹ ï/ï';
  SGZ.Cells[1,0]:='Íîìåð';
  SGZ.Cells[2,0]:='Äàòà';
  SGZ.Cells[3,0]:='Èíäåêñ';
  SGZ.Cells[4,0]:='Òåêñò(êðàòêî)';
  SGZ.ColWidths[4]:=400;

  if DBLkpCB_Dep4.KeyValue<>-1 then
  begin
            s:='select   IND,   PERIOD   from   TYPECASES   where   DEP='   +
IntToStr(DBLkpCB_Dep4.KeyValue);
    // Showmessage(s);
    with DM.IBQuery1 do
    begin
      //SQL.Clear;   SQL.Add('COMMIT;');   ExecSQL;
      SQL.Clear;
      SQL.Add(s);
      Open;

      First;
      while not EOF do
      begin
      if Fields[1].AsString='ïîñò'
      then TMP[Fields[0].AsiNTEGER]:=200
      else TMP[Fields[0].AsiNTEGER]:=Fields[1].asInteger;
      Next;
      end;
    end;

     s:='select CASE_NUM, CASE_DATE, DEP, IND, CASE_TEXT from CASES where (DEP='
+ IntToStr(DBLkpCB_Dep4.KeyValue) + ') and (CASE_DATE between ''' + ME21.Text +
''' and ''' + ME22.Text + ''') order by CASE_DATE';
    with DM.IBQuery1 do
    begin
      SQL.Clear;
      SQL.Add(s);
      Open;

      First; i:=1;
      while not EOF do
      begin
      if TMP[Fields[3].AsINTEGER]>199 then
      begin
        SGZ.RowCount:=i+1;
        SGZ.Cells[0,i] :=IntToStr(i);
        SGZ.Cells[1,i] :=Fields[0].AsString;
        SGZ.Cells[2,i] :=Fields[1].AsString;
        SGZ.Cells[3,i] :=Fields[2].AsString + '-' + Fields[3].AsString;;
        SGZ.Cells[4,i] :=COpy(Fields[4].AsString,1,128);
        inc(i);
      end;
      Next;
      end;
    end;
  end
  else begin
       s:='select IND, PERIOD, DEP from TYPECASES';
       with DM.IBQuery1 do
       begin
         //SQL.Clear;   SQL.Add('COMMIT;');   ExecSQL;
         SQL.Clear;
```

```
        SQL.Add(s);
        Open;
        First;
        while not EOF do
        begin
          if Fields[1].AsString='ïîñò'
          then TMP2[Fields[0].AsInteger, Fields[2].asInteger]:=200
          else TMP2[Fields[0].AsInteger, Fields[2].asInteger]:=
                                                  Fields[1].asInteger;
          Next;
        end;
      end;

      s:='select  CASE_NUM,  CASE_DATE,  DEP,  IND,  CASE_TEXT  from  CASES  where
(CASE_DATE between ''' + ME21.Text + ''' and ''' + ME22.Text + ''') order by
CASE_DATE';
       with DM.IBQuery1 do
       begin
         SQL.Clear;
         SQL.Add(s);
         Open;
         First;
         i:=1;
         while not EOF do
         begin
           if TMP2[Fields[3].AsINTEGER,Fields[2].asInteger]>199 then
           begin
             SGZ.RowCount:=i+1;
             SGZ.Cells[0,i] :=IntToStr(i);
             SGZ.Cells[1,i] :=Fields[0].AsString;
             SGZ.Cells[2,i] :=Fields[1].AsString;
             SGZ.Cells[3,i] :=Fields[2].AsString + '-' + Fields[3].AsString;;
             SGZ.Cells[4,i] :=COpy(Fields[4].AsString,1,128);
             inc(i);
           end;
           Next;
         end;
       end;
       end;
end;

procedure TFMain.SB6Click(Sender: TObject);
  var s:string;
      i:integer;
      TMP:array[1..100] of integer;
      tmp2:array[1..100,1..100] of integer;
begin
  SGZ.RowCount:=2;
  SGZClear;

  SGZ.Tag :=6; //pokazatel chto budet print case
  SGZ.ColCount := 5;
  SGZ.Cells[0,0]:='¹ ï/ï';
  SGZ.Cells[1,0]:='Íîìåð';
  SGZ.Cells[2,0]:='Äàòà';
  SGZ.Cells[3,0]:='Èíäåêñ';
  SGZ.Cells[4,0]:='Òåêñò(êðàòêî)';
  SGZ.ColWidths[4]:=400;

  if DBLkpCB_Dep4.KeyValue<>-1 then
  begin
```

```
            s:='select   IND,   PERIOD   from   TYPECASES   where   DEP='   +
IntToStr(DBLkpCB_Dep4.KeyValue);
    with DM.IBQuery1 do
    begin
      SQL.Clear;
      SQL.Add(s);
      Open;
      First;
      while not EOF do
      begin
      if Fields[1].AsString='ïîñò'
      then TMP[Fields[0].AsiNTEGER]:=200
      else TMP[Fields[0].AsiNTEGER]:=Fields[1].asInteger;
      Next;
      end;
    end;
     s:='select CASE_NUM, CASE_DATE, DEP, IND, CASE_TEXT from CASES where (DEP='
+ IntToStr(DBLkpCB_Dep4.KeyValue) + ') and (CASE_DATE between ''' + ME21.Text +
''' and ''' + ME22.Text + ''') order by CASE_DATE';
    with DM.IBQuery1 do
    begin
      SQL.Clear;
      SQL.Add(s);
      Open;
      First;
      i:=1;
      while not EOF do
      begin
      if (TMP[Fields[3].AsINTEGER]>10) and (tmp[Fields[3].AsINTEGER]<199) then
      begin
        SGZ.RowCount:=i+1;
        SGZ.Cells[0,i] :=IntToStr(i);
        SGZ.Cells[1,i] :=Fields[0].AsString;
        SGZ.Cells[2,i] :=Fields[1].AsString;
        SGZ.Cells[3,i] :=Fields[2].AsString + '-' + Fields[3].AsString;;
        SGZ.Cells[4,i] :=COpy(Fields[4].AsString,1,128);
        inc(i);
      end;
      Next;
      end;
    end;
  end
  else begin
       s:='select IND, PERIOD, DEP from TYPECASES';
       with DM.IBQuery1 do
       begin
         SQL.Clear;
         SQL.Add(s);
         Open;
         First;
         while not EOF do
         begin
           if Fields[1].AsString='ïîñò'
           then TMP2[Fields[0].AsiNTEGER, Fields[2].asInteger]:=200
                                                                    else
TMP2[Fields[0].AsiNTEGER,Fields[2].asInteger]:=Fields[1].asInteger;
           Next;
         end;
       end;
```

```
       s:='select CASE_NUM, CASE_DATE, DEP, IND, CASE_TEXT from CASES where
(CASE_DATE between ''' + ME21.Text + ''' and ''' + ME22.Text + ''') order by
CASE_DATE';
       with DM.IBQuery1 do
       begin
         SQL.Clear;
         SQL.Add(s);
         Open;
         First;
         i:=1;
         while not EOF do
         begin
                  if  (TMP2[Fields[3].AsINTEGER,  Fields[2].asInteger]>10)  and
(tmp2[Fields[3].AsINTEGER,Fields[2].asInteger]<199)
           then begin
                SGZ.RowCount:=i+1;
                SGZ.Cells[0,i] :=IntToStr(i);
                SGZ.Cells[1,i] :=Fields[0].AsString;
                SGZ.Cells[2,i] :=Fields[1].AsString;
                SGZ.Cells[3,i] :=Fields[2].AsString + '-' + Fields[3].AsString;;
                SGZ.Cells[4,i] :=COpy(Fields[4].AsString,1,128);
                inc(i);
              end;
           Next;
         end;
       end;
     end;
end;

procedure TFMain.SB7Click(Sender: TObject);
  var s:string;
      i:integer;
      TMP:array[1..100] of integer;
      tmp2:array[1..100,1..100] of integer;
begin
   SGZ.RowCount:=2;
   SGZClear;

   SGZ.Tag :=7; //pokazatel chto budet print case
   SGZ.ColCount := 5;
   SGZ.Cells[0,0]:='¹ ï/ï';
   SGZ.Cells[1,0]:='Íîìåð';
   SGZ.Cells[2,0]:='Äàòà';
   SGZ.Cells[3,0]:='Èíäåêñ';
   SGZ.Cells[4,0]:='Òåêñò(êðàòêî)';
   SGZ.ColWidths[4]:=400;

  if DBLkpCB_Dep4.KeyValue<>-1 then
  begin
             s:='select   IND,   PERIOD   from   TYPECASES   where   DEP='   +
IntToStr(DBLkpCB_Dep4.KeyValue);
    with DM.IBQuery1 do
    begin
      SQL.Clear;
      SQL.Add(s);
      Open;
      First;
      while not EOF do
      begin
      if Fields[1].AsString='ïîñò'
      then TMP[Fields[0].AsiNTEGER]:=200
      else TMP[Fields[0].AsiNTEGER]:=Fields[1].asInteger;
```

```
      Next;
      end;
         s:='select CASE_NUM, CASE_DATE, DEP, IND, CASE_TEXT from CASES where
(DEP=' + IntToStr(DBLkpCB_Dep4.KeyValue) + ') and (CASE_DATE between ''' +
ME21.Text + ''' and ''' + ME22.Text + ''') order by CASE_DATE';

    with DM.IBQuery1 do
    begin
      SQL.Clear;
      SQL.Add(s);
      Open;
      First;
      i:=1;
      while not EOF do
      begin
      if TMP[Fields[3].AsINTEGER]<=10 then
      begin
        SGZ.RowCount:=i+1;
        SGZ.Cells[0,i] :=IntToStr(i);
        SGZ.Cells[1,i] :=Fields[0].AsString;
        SGZ.Cells[2,i] :=Fields[1].AsString;
        SGZ.Cells[3,i] :=Fields[2].AsString + '-' + Fields[3].AsString;;
        SGZ.Cells[4,i] :=COpy(Fields[4].AsString,1,128);
        inc(i);
      end;
      Next;
      end;
    end;
  end
  else begin
       s:='select IND, PERIOD, DEP from TYPECASES';
       with DM.IBQuery1 do
       begin
         SQL.Clear;
         SQL.Add(s);
         Open;
         First;
         while not EOF do
         begin
           if Fields[1].AsString='ïîñò'
           then TMP2[Fields[0].AsiNTEGER,Fields[2].AsINTEGER]:=200
                                                                    else
TMP2[Fields[0].AsiNTEGER,Fields[2].AsINTEGER]:=Fields[1].asInteger;
           Next;
         end;
        s:='select CASE_NUM, CASE_DATE, DEP, IND, CASE_TEXT Ffrom CASES where
(CASE_DATE between ''' + ME21.Text + ''' and ''' + ME22.Text + ''') order by
CASE_DATE';
       with DM.IBQuery1 do
       begin
         SQL.Clear;
         SQL.Add(s);
         Open;
         First;
         i:=1;
         while not EOF do
         begin
           if TMP2[Fields[3].AsINTEGER,Fields[2].AsINTEGER]<=10 then
           begin
             SGZ.RowCount:=i+1;
             SGZ.Cells[0,i] :=IntToStr(i);
             SGZ.Cells[1,i] :=Fields[0].AsString;
```

```
            SGZ.Cells[2,i] :=Fields[1].AsString;
            SGZ.Cells[3,i] :=Fields[2].AsString + '-' + Fields[3].AsString;;
            SGZ.Cells[4,i] :=Copy(Fields[4].AsString,1,128);
            inc(i);
          end;
          Next;
        end;
      end;
      end;
end;  //sdelat' bez DEP


{=======ÇÀÊËÀÄÊÀ <<<<<<<<ÏÎÈÑÊ ÄÎÊÓÌÅÍÒÀ>>>>>>>>>>>}
procedure TFMain.Button1Click(Sender: TObject);
  var s:string;
      i:integer;
begin
  SGFind.RowCount :=2;
  for i:=0 to SGFind.ColCount-1 do
  begin
    SGFind.Cells[i,1] :='';
    SGFind.ColWidths[i]:=100;
  end;

  if CB1.ItemIndex = 0 then
  begin
    s:='Select * from DECREES';
    if RB1.Checked = True then s:=s + ' where DECREE_NUM=''' + E_FindNum.Text +
''';';
      if RB2.Checked = True then s:=s + ' WHERE DECREE_TEXT like ''%' +
E_FindText.Text + '%'' order by DECREE_NUM';
    if RB3.Checked = True then s:=s + ' WHERE DATE_=''' + ME_FindDate.Text + '''
order by DECREE_NUM';
    SGFind.ColCount := 4;
    SGFind.Cells[0,0]:='¹ ï/ï';
    SGFind.Cells[1,0]:='Íîìåð';
    SGFind.Cells[2,0]:='Äàòà';
    SGFind.Cells[3,0]:='Òåêñò(êðàòêî)';
    SGFind.ColWidths[3]:=400;
  end;

  if CB1.ItemIndex = 1 then
  begin
    s:='Select * from CASES';
    if RB1.Checked = True then s:=s + ' where CASE_NUM=''' + E_FindNum.Text +
'''';
      if RB2.Checked = True then s:=s + ' where CASE_TEXT LIKE ''%' +
E_FindText.Text + '%'' order by CASE_NUM';
    if RB3.Checked = True then s:=s + ' where CASE_DATE=''' + ME_FindDate.Text +
''' order by CASE_NUM';
    SGFind.ColCount := 5;
    SGFind.Cells[0,0]:='¹ ï/ï';
    SGFind.Cells[1,0]:='Íîìåð';
    SGFind.Cells[2,0]:='Äàòà';
    SGFind.Cells[3,0]:='Èíäåêñ';
    SGFind.Cells[4,0]:='Òåêñò(êðàòêî)';
    SGFind.ColWidths[4]:=400;
  end;

  if CB1.ItemIndex = 2 then
  begin
    s:='Select * from INCORS';
```

```
    if RB1.Checked = True then s:=s + ' where ID=' + E_FindNum.Text;
       if RB2.Checked = True then s:=s + ' where INCOR_TEXT like ''%' +
E_FindText.Text + '%'' order by ID';
    if RB3.Checked = True then s:=s + ' where DATE1=''' + ME_FindDate.Text + '''
order by ID';
    SGFind.ColCount := 8;
    SGFind.Cells[0,0]:='¹ ï/ï';
    SGFind.Cells[1,0]:='Äàòà ïîëó÷.';
    SGFind.Cells[2,0]:='Îò êîãî';
    SGFind.Cells[3,0]:='Íîìåð âõîä.';
    SGFind.Cells[4,0]:='Äàòà âõîä';
    SGFind.Cells[5,0]:='Ñîäåðæàíèå';
    SGFind.Cells[6,0]:='Èñïîëíèòåëü';
    SGFind.Cells[7,0]:='Îòìåòêà';
    SGFind.ColWidths[5]:=400; //ShowMEssage(s);
  end;

  if CB1.ItemIndex = 3 then
  begin
    s:='Select * from OUTCORS';
    if RB1.Checked = True then s:=s+' where ID=' + E_FindNum.Text;
       if RB2.Checked = True then s:=s+' where OUTCOR_TEXT LIKE ''%' +
E_FindText.Text + '%'' order by ID';
    if RB3.Checked = True then s:=s+' where  DATE1=''' + ME_FindDate.Text + '''
order by ID';
    SGFind.ColCount := 6;
    SGFind.Cells[0,0]:='¹ ï/ï';
    SGFind.Cells[1,0]:='Íîìåð';
    SGFind.Cells[2,0]:='Äàòà îòïðàâ.';
    SGFind.Cells[3,0]:='Êóäà, êîìó';
    SGFind.Cells[4,0]:='Ñîäåðæàíèå (êðàòêî)';
    SGFind.Cells[5,0]:='Êîïèÿ (Ïîäðàçäåëåíèå)';
    SGFind.ColWidths[4]:=400;
  end;

  with DM.IBQuery1 do
  begin
    SQL.Clear;
    SQL.Add(s);
    Open;
    First; i:=1;
    while not EOF do
    begin
      SGFind.RowCount:=i+1;
      SGFind.Cells[0,i] :=IntToStr(i);

      if CB1.ItemIndex = 0 then //DECREES
      begin
      SGFind.Cells[1,i] :=Fields[1].AsString;
      SGFind.Cells[2,i] :=Fields[4].AsString;
      SGFind.Cells[3,i] :=Copy(Fields[3].AsString,1,128);
      end;

      if CB1.ItemIndex = 1 then //CASES
      begin
      SGFind.Cells[1,i] :=Fields[1].AsString;
      SGFind.Cells[2,i] :=Fields[4].AsString;
      SGFind.Cells[3,i] :=Fields[2].AsString + '-' + Fields[3].AsString;;
      SGFind.Cells[4,i] :=Copy(Fields[5].AsString,1,128);
      end;

      if CB1.ItemIndex = 2 then //VHOD KORESP
```

```
      begin
      SGFind.Cells[1,i] :=Fields[1].AsString;
      SGFind.Cells[2,i] :=Fields[2].AsString;
      SGFind.Cells[3,i] :=Fields[3].AsString;
      SGFind.Cells[4,i] :=Fields[4].AsString;
      SGFind.Cells[5,i] :=Copy(Fields[5].AsString,1,128);
      SGFind.Cells[6,i] :=Fields[6].AsString;;
      SGFind.Cells[7,i] :=Fields[7].AsString;;
      end;

     if CB1.ItemIndex = 3 then //ISHOD KORESP
     begin
       SGFind.Cells[1,i] :=Fields[0].AsString;
       SGFind.Cells[2,i] :=Fields[1].AsString;
       SGFind.Cells[3,i] :=Fields[2].AsString;
       SGFind.Cells[4,i] :=Copy(Fields[3].AsString,1,128);
       SGFind.Cells[5,i] :=Fields[4].AsString;
     end;
     inc(i);
     Next;
    end;
    //SQL.Clear;   SQL.Add('COMMIT;');  ExecSQL;
  end;
end;

{====================PRINT======================}
procedure TFMain.Button5Click(Sender: TObject);
  var range1,x,y: OleVariant;
      FileName,FileName1: OleVariant;
      i,j,m,w:integer; k:REAL;
begin    {!!!   Only for MS Word 2000-XP}
  FileName1:=PathProg + '\report\dep.doc';
  FileName :=PathProg + '\report\dep.tmp';

  if FileExists(FileName)
  then DeleteFile(FileName);

  CopyFile(pchar(PathProg+'\report\dep.doc'),pchar(PathProg +
                                                   '\report\dep.tmp'),true);

  WordInit(FileName); //Initialize

  WordGotoBookmark('Director');
  WordInsertText(DIrectorName);

  DM.IBQ_Dep.Last;

  i:=DM.IBQ_Dep.RecordCount;
  j:=3; //KOlichestvo strok i stolbcov

  x:=Length(WordDocument1.Range.Text);
  y:=x-1;
      WordDocument1.Tables.Add(WordDocument1.Range(y,x),   i,   j,   EmptyParam,
EmptyParam);

  DM.IBQ_Dep.First;
  m:=1;
  While not DM.IBQ_Dep.Eof do
  begin
    WordDocument1.Tables.Item(1).Cell(m, 1).Range.Text:=DM.IBQ_DepID.AsString;
    WordDocument1.Tables.Item(1).Cell(m, 2).Range.Text:=DM.IBQ_DepDEP.AsString;
    WordDocument1.Tables.Item(1).Cell(m, 3).Range.Text:=DM.IBQ_DepINFO.AsString;
```

```
    DM.IBQ_Dep.Next;
    inc(m);
  end;

  range1 := WordDocument1.Range;
  range1.InsertAfter(#10#13#10#13+DateToStr(date));

  TableF(1);
    w:=Trunc(WordDocument1.PageSetup.PageWidth);
    k:=w/21;
    WordDocument1.Tables.Item(1).Columns.Item(1).Width:=Round((w-5*k)*0.1);
    WordDocument1.Tables.Item(1).Columns.Item(2).Width:=Round((W-5*k)*0.62);
    WordDocument1.Tables.Item(1).Columns.Item(3).Width:=Round((W-5*k)*0.28);

  WordDocument1.PrintOut;
  WordEnd(FileName);
end;

procedure TFMain.Timer1Timer(Sender: TObject);// print TYPECASE
begin
 WordApplication1.Disconnect;
end;

procedure TFMain.B_PrintTypeCasesClick(Sender: TObject);
var
  range1,x,y: OleVariant;
  FileName,FileName1: OleVariant;
  i,j,m,w:integer;
  k:real;
begin    {!!!   Only for MS Word 2000- XP}
  FileName1:=PathProg + '\report\typecase.doc';
  FileName :=PathProg + '\report\typecase.tmp';

  if FileExists(FileName) then DeleteFile(FileName);
       CopyFile(pchar(PathProg+    '\report\typecase.doc'),    pchar(PathProg    +
'\report\typecase.tmp'),true);

  WordInit(FileName); //Initialize

  WordGotoBookmark('Director');
  WordInsertText(DIrectorName);

  DM.IBQ_TypeCases.Last;

  i:=DM.IBQ_TypeCases.RecordCount;
  j:=5; //KOlichestvo strok i stolbcov

  x:=Length(WordDocument1.Range.Text);
  y:=x-1;
      WordDocument1.Tables.Add(WordDocument1.Range(y,x),   i,   j,   EmptyParam,
EmptyParam);

  DM.IBQ_TypeCases.First;
  m:=1;
  While not DM.IBQ_TypeCases.Eof do
  begin
                                          WordDocument1.Tables.Item(1).Cell(m,
1).Range.Text:=DM.IBQ_TypeCasesIND.AsString;
                                          WordDocument1.Tables.Item(1).Cell(m,
2).Range.Text:=DM.IBQ_TypeCasesNAMECASE.AsString;
                                          WordDocument1.Tables.Item(1).Cell(m,
3).Range.Text:=DM.IBQ_TypeCasesCOUNTPART.AsString;
```

```
                                                WordDocument1.Tables.Item(1).Cell(m,
4).Range.Text:=DM.IBQ_TypeCasesPER_NUM.AsString;
                                                WordDocument1.Tables.Item(1).Cell(m,
5).Range.Text:=DM.IBQ_TypeCasesINFO.AsString;
    DM.IBQ_TypeCases.Next;
    inc(m);
  end;

  range1 := WordDocument1.Range;
  range1.InsertAfter(#10#13#10#13+DateToStr(date));

  TableF(1);
     //Editing Table.Cols
    w:=Trunc(WordDocument1.PageSetup.PageWidth);
    k:=w/21;
    WordDocument1.Tables.Item(1).Columns.Item(1).Width:=Round((w-5*k)*0.1);
    WordDocument1.Tables.Item(1).Columns.Item(2).Width:=Round((W-5*k)*0.52);
    WordDocument1.Tables.Item(1).Columns.Item(3).Width:=Round((W-5*k)*0.08);
    WordDocument1.Tables.Item(1).Columns.Item(4).Width:=Round((W-5*k)*0.15);
    WordDocument1.Tables.Item(1).Columns.Item(5).Width:=Round((W-5*k)*0.15);

  WordDocument1.PrintOut;
  WordEnd(FileName); //Çàâåðøåíèå ðàáîòû ñ Word
end;

procedure TFMain.B_PrintSaveDecreeClick(Sender: TObject); //print NAKAZ
  var FileName: OleVariant;
begin
  FileName:=PathProg + '\report\prikaz.tmp';
  if FileExists(FileName) then DeleteFile(FileName);
      CopyFile(pchar(PathProg   +   '\report\prikaz.doc'),   pchar(PathProg   +
'\report\prikaz.tmp'), true);

  WordInit(FileName); //Initialize

  {Âñòàâëÿåì òåêñò ïðèêàçà}
  WordGotoBookmark('date');
  WordInsertText(ME_DateDecree.Text);
  WordGotoBookmark('nomer');
  WordInsertText(E_NumDecree.Text);
  WordGotoBookmark('text');
  WordInsertText(M_DecreeText.Text);
  WordGotoBookmark('director');
  WordInsertText(DirectorName);

  WordDocument1.PrintOut;
  WordEnd(FileName); //Çàâåðøåíèå ðàáîòû ñ Word
end;

procedure TFMain.B_PrintCaseClick(Sender: TObject);  //print Case
  var FileName: OleVariant;
begin
  FileName:=PathProg + '\report\case.tmp';
  if FileExists(FileName)
  then DeleteFile(FileName);

       CopyFile(pchar(PathProg   +   '\report\case.doc'),   pchar(PathProg   +
'\report\case.tmp'),true);

  WordInit(FileName);

  WordGotoBookmark('dep');
```

```
  WordInsertText(DBLkpCB_Dep2.Text);

  WordGotoBookmark('nomer');
  WordInsertText(E_NumCase.Text);

  WordGotoBookmark('ind');
  WordInsertText(CB_IndexCase.Text);

  WordGotoBookmark('title');
  WordInsertText(tn3[CB_IndexCase.ItemIndex]);

  WordGotoBookmark('text');
  WordInsertText(M_InfoCase.Text);

  WordGotoBookmark('date');
  WordInsertText(ME_DateCase.Text);

  WordDocument1.PrintOut;

  WordEnd(FileName);
end;

procedure TFMain.SB1Click(Sender: TObject);
  var FileName: OleVariant;
begin
  FileName:=PathProg + '\report\blank.doc';
  WordInit(FIleNAme);
  WordDocument1.Connect;

  WordApplication1.Options.CheckSpellingAsYouType:=True;
  WordApplication1.Options.CheckGrammarAsYouType:=True;

  WordApplication1.Visible:=true;
end;

procedure TFMain.Button18Click(Sender: TObject);
begin
  case SGZ.Tag of
    2 : print2;
    3 : print3;
    4 : print4;
    5 : print5;
    6 : print6;
    7 : print7;
  end;
end;

procedure TFMain.print2;
  var FileName,x,y: OleVariant;
      i,j,m,n,w:integer;
      k:real; s:string;
begin    {!!!   Only for MS Word 2000-XP}
  FileName:=PathProg + '\report\rep1.tmp';
  if FileExists(FileName)
  then DeleteFile(FileName);
       CopyFile(pchar(PathProg    +    '\report\rep1.doc'),    pchar(PathProg    +
'\report\rep1.tmp'), true);

  WordInit(FileName);

  {TExt}
  WordGotoBookmark('date');
```

```
  WordInsertText(DateToStr(date));
  WordGotoBookmark('type');
  WordInsertText(' ÏÐÈÊÀÇÀÌ');
  WordGotoBookmark('param');
  s:='Ïåðèîä:';
  if ME21.Text='  .  .    '
  then s:=s+' _ñ íà÷àëà âåäåíèÿ áàçû_   '
  else s:=s+' ñ _'+ ME21.Text + '_  ';
  s:=s+' ïî  _' + ME22.Text + '_  ';
  WordInsertText(s);

  i:=SGZ.RowCount; j:=4; //KOlichestvo strok i stolbcov
  x:=Length(WordDocument1.Range.Text); y:=x-1;
      WordDocument1.Tables.Add(WordDocument1.Range(y,x),   i,   j,   EmptyParam,
EmptyParam);

  //formirovanie table
  for m:=1 to i do
  for n:=1 to j do
    WordDocument1.Tables.Item(1).Cell(m,n).Range.Text:=SGZ.Cells[n-1,m-1];

  TableF(1);    //ôîðìàòèðîâàíèå òàáëèöû
  w:=Trunc(WordDocument1.PageSetup.PageWidth);
  k:=w/21;
  WordDocument1.Tables.Item(1).Columns.Item(1).Width:=Round((w-5*k)*0.1);
  WordDocument1.Tables.Item(1).Columns.Item(2).Width:=Round((W-5*k)*0.17);
  WordDocument1.Tables.Item(1).Columns.Item(3).Width:=Round((W-5*k)*0.15);
  WordDocument1.Tables.Item(1).Columns.Item(4).Width:=Round((W-5*k)*0.58);

  if Button2.Tag = 0
  then WordDocument1.PrintOut;
  WordEnd(FileName);
end;

procedure TFMain.print3; //ïî âðåìåíè ñîçäàíèÿ
  var FileName: OleVariant;
      s:string;
begin    {!!!   Only for MS Word 2000-XP}
  FileName:=PathProg + '\report\rep2.tmp';

  if FileExists(FileName)
  then DeleteFile(FileName);

       CopyFile(pchar(PathProg   +   '\report\rep2.doc'),   pchar(PathProg   +
'\report\rep2.tmp'),true);

  WordInit(FileName); //Initialize

  {TExt}
  WordGotoBookmark('date');
  WordInsertText(DateToStr(date));
  WordGotoBookmark('period');

  If ME21.Text='  .  .    '
  then s:=s + ' _ñ íà÷àëà âåäåíèÿ áàçû_   '
  else s:=s + ' ñ _' + ME21.Text + '_  ';

  s:=s + ' ïî  _' + ME22.Text + '_  ';
  WordInsertText(s);

  WordGotoBookmark('dep');
```

```
        WordInsertText(IntToStr(DBLkpCB_Dep4.KeyValue)    +    '         '    +
tn2[Integer(DBLkpCB_Dep4.KeyValue)]);

  WordGotoBookmark('r1');
  WordInsertText(SGZ.Cells[1,1]);

  WordGotoBookmark('r2');
  WordInsertText(SGZ.Cells[1,2]);

  WordGotoBookmark('r3');
  WordInsertText(SGZ.Cells[1,3]);

  WordGotoBookmark('r4');
  WordInsertText(SGZ.Cells[1,4]);

  if Button2.Tag = 0 then WordDocument1.PrintOut;
  WordEnd(FileName); //Çàâåðøåíèå ðàáîòû ñ Word
end;

procedure TFMain.print4; // spravy
var
  FileName,x,y: OleVariant;
  i,j,m,n,w:integer; k:REAL; s:string;
begin    {!!!   Only for MS Word 2000-XP}
  FileName:=PathProg+ '\report\rep1.tmp';
  if FileExists(FileName)
  then DeleteFile(FileName);

        CopyFile(pchar(PathProg    +    '\report\rep1.doc'),pchar(PathProg    +
'\report\rep1.tmp'),true);

  WordInit(FileName); //Initialize

  {TExt}
  WordGotoBookmark('date');
  WordInsertText(DateToStr(date));
  WordGotoBookmark('type');
  WordInsertText(' ÄÅËÀÌ');
  WordGotoBookmark('param');
  s:='Ïîäðàçäåëåíèå: ';
  if DBLkpCB_Dep4.KeyValue=-1
  then s:=s + '_âñå_;   '
  else s:=s + '_' + IntToStr(DBLkpCB_Dep4.KeyValue) + '_;   ';

  If ME21.Text='  .  .    '
  then s:=s + 'Ïåðèîä: _ñ íà÷àëà âåäåíèÿ áàçû_   '
  else s:=s+'Ïåðèîä: ñ _' + ME21.Text + '_  ';
  s:=s+' ïî  _' + ME22.Text + '_  ';
  WordInsertText(s);

  i:=SGZ.RowCount; j:=5; //KOlichestvo strok i stolbcov
  x:=Length(WordDocument1.Range.Text); y:=x-1;
      WordDocument1.Tables.Add(WordDocument1.Range(y,x),   i,   j,   EmptyParam,
EmptyParam);

  for m:=1 to i do
  for n:=1 to j do
    WordDocument1.Tables.Item(1).Cell(m,n).Range.Text:=SGZ.Cells[n-1,m-1];

  TableF(1);    //ôîðìàòèðîâàíèå òàáëèöû
  w:=Trunc(WordDocument1.PageSetup.PageWidth);
  k:=w/21;
```

```
  WordDocument1.Tables.Item(1).Columns.Item(1).Width:=Round((w-5*k)*0.08);
  WordDocument1.Tables.Item(1).Columns.Item(2).Width:=Round((W-5*k)*0.12);
  WordDocument1.Tables.Item(1).Columns.Item(3).Width:=Round((W-5*k)*0.15);
  WordDocument1.Tables.Item(1).Columns.Item(4).Width:=Round((W-5*k)*0.15);
  WordDocument1.Tables.Item(1).Columns.Item(5).Width:=Round((W-5*k)*0.5);

  if Button2.Tag = 0
  then WordDocument1.PrintOut;
  WordEnd(FileName);
end;


procedure TFMain.print5;
var
  FileName,x,y: OleVariant;
  i,j,m,n,w:integer; k:REAL; s:string;
begin    {!!!   Only for MS Word 2000-XP}
  FileName:=PathProg + '\report\rep3.tmp';
  if FileExists(FileName)
  then DeleteFile(FileName);
       CopyFile(pchar(PathProg    +    '\report\rep3.doc'),    pchar(PathProg    +
'\report\rep3.tmp'), true);

  WordInit(FileName); //Initialize

  {TExt}
  WordGotoBookmark('date');
  WordInsertText(DateToStr(date));
  WordGotoBookmark('type');
  WordInsertText('Ïîñòîÿííîãî õðàíåíèÿ');
  WordGotoBookmark('param');
  s:='Ïîäðàçäåëåíèå: ';
  if DBLkpCB_Dep4.KeyValue=-1
  then s:=s + '_âñå_;   '
  else s:=s + '_' + IntToStr(DBLkpCB_Dep4.KeyValue) + '_;   ';

  If ME21.Text='  .  .    '
  then s:=s + 'Ïåðèîä: _ñ íà÷àëà âåäåíèÿ áàçû_   '
  else s:=s + 'Ïåðèîä: ñ _' + ME21.Text + '_  ';
  s:=s + ' ïî  _' + ME22.Text + '_  ';
  WordInsertText(s);

  i:=SGZ.RowCount; j:=5; //KOlichestvo strok i stolbcov
  x:=Length(WordDocument1.Range.Text);
  y:=x-1;
      WordDocument1.Tables.Add(WordDocument1.Range(y,x),   i,   j,   EmptyParam,
EmptyParam);

  //formirovanie table
  for m:=1 to i do
  for n:=1 to j do
    WordDocument1.Tables.Item(1).Cell(m,n).Range.Text:=SGZ.Cells[n-1,m-1];

  TableF(1);
    w:=Trunc(WordDocument1.PageSetup.PageWidth);
    k:=w/21;
    WordDocument1.Tables.Item(1).Columns.Item(1).Width:=Round((w-5*k)*0.08);
    WordDocument1.Tables.Item(1).Columns.Item(2).Width:=Round((W-5*k)*0.12);
    WordDocument1.Tables.Item(1).Columns.Item(3).Width:=Round((W-5*k)*0.15);
    WordDocument1.Tables.Item(1).Columns.Item(4).Width:=Round((W-5*k)*0.15);
    WordDocument1.Tables.Item(1).Columns.Item(5).Width:=Round((W-5*k)*0.5);
```

```
  if Button2.Tag = 0 then WordDocument1.PrintOut;
  WordEnd(FileName); //Çàâåðøåíèå ðàáîòû ñ Word
end;

procedure TFMain.print6;
var
  FileName,x,y: OleVariant;
  i,j,m,n,w:integer; k:REAL; s:string;
begin    {!!!   Only for MS Word 2000-XP}
  FileName:=PathProg + '\report\rep3.tmp';
  if FileExists(FileName) then DeleteFile(FileName);
        CopyFile(pchar(PathProg    +    '\report\rep3.doc'),pchar(PathProg    +
'\report\rep3.tmp'),true);

  WordInit(FileName);
  WordGotoBookmark('date');
  WordInsertText(DateToStr(date));
  WordGotoBookmark('type');
  WordInsertText('Äëèòåëüíîñò õðàíåíèÿ');
  WordGotoBookmark('param');
  s:='Ïîäðàçäåë: ';
  if DBLkpCB_Dep4.KeyValue=-1
  then s:=s + '_âñ³_;   '
  else s:=s + '_' + IntToStr(DBLkpCB_Dep4.KeyValue) + '_;   ';

  if ME21.Text='  .  .    '
  then s:=s + 'Ïåðèîä: _ñ íà÷àëà âåäåíèÿ áàçû_   '
  else s:=s + 'Ïåðèîä: ñ _' + ME21.Text + '_  ';

  s:=s+' ïî  _' + ME22.Text+'_  ';
  WordInsertText(s);

  i:=SGZ.RowCount; j:=5; //KOlichestvo strok i stolbcov
  x:=Length(WordDocument1.Range.Text); y:=x-1;
      WordDocument1.Tables.Add(WordDocument1.Range(y,x),   i,   j,   EmptyParam,
EmptyParam);

     //formirovanie table
    for m:=1 to i do
    for n:=1 to j do
      WordDocument1.Tables.Item(1).Cell(m,n).Range.Text:=SGZ.Cells[n-1,m-1];

  TableF(1);
    w:=Trunc(WordDocument1.PageSetup.PageWidth);
    k:=w/21;
    WordDocument1.Tables.Item(1).Columns.Item(1).Width:=Round((w-5*k)*0.08);
    WordDocument1.Tables.Item(1).Columns.Item(2).Width:=Round((W-5*k)*0.12);
    WordDocument1.Tables.Item(1).Columns.Item(3).Width:=Round((W-5*k)*0.15);
    WordDocument1.Tables.Item(1).Columns.Item(4).Width:=Round((W-5*k)*0.15);
    WordDocument1.Tables.Item(1).Columns.Item(5).Width:=Round((W-5*k)*0.5);

  if Button2.Tag = 0 then WordDocument1.PrintOut;
  WordEnd(FileName);
end;


procedure TFMain.print7;
var
  FileName,x,y: OleVariant;
  i,j,m,n,w:integer; k:REAL; s:string;
begin    {!!!   Only for MS Word 2000-XP}
  FileName:=PathProg + '\report\rep3.tmp';
```

```
  if FileExists(FileName)
  then DeleteFile(FileName);
       CopyFile(pchar(PathProg    +    '\report\rep3.doc'),    pchar(PathProg    +
'\report\rep3.tmp'), true);

  WordInit(FileName); //Initialize

  {TExt}
  WordGotoBookmark('date');
  WordInsertText(DateToStr(date));
  WordGotoBookmark('type');
  WordInsertText('Âðåìåííîãî õðàíåíèÿ');
  WordGotoBookmark('param');
  s:='Ïîäðàçäåëåíèå: ';

  if DBLkpCB_Dep4.KeyValue=-1
  then s:=s + '_âñå_;   '
  else s:=s + '_' + IntToStr(DBLkpCB_Dep4.KeyValue) + '_;   ';

  If ME21.Text='  .  .    '
  then s:=s + 'Ïåðèîä: _ñ íà÷àëà âåäåíèÿ áàçû_   '
  else s:=s + 'Ïåðèîä: ñ _' + ME21.Text + '_  ';
  s:=s+' ïî  _' + ME22.Text + '_  ';
  WordInsertText(s);


  i:=SGZ.RowCount; j:=5; //KOlichestvo strok i stolbcov
  x:=Length(WordDocument1.Range.Text);
  y:=x-1;
      WordDocument1.Tables.Add(WordDocument1.Range(y,x),   i,   j,   EmptyParam,
EmptyParam);

     //formirovanie table
    for m:=1 to i do
    for n:=1 to j do
      WordDocument1.Tables.Item(1).Cell(m,n).Range.Text:=SGZ.Cells[n-1,m-1];

  TableF(1);
    w:=Trunc(WordDocument1.PageSetup.PageWidth);
    k:=w/21;
    WordDocument1.Tables.Item(1).Columns.Item(1).Width:=Round((w-5*k)*0.08);
    WordDocument1.Tables.Item(1).Columns.Item(2).Width:=Round((W-5*k)*0.12);
    WordDocument1.Tables.Item(1).Columns.Item(3).Width:=Round((W-5*k)*0.15);
    WordDocument1.Tables.Item(1).Columns.Item(4).Width:=Round((W-5*k)*0.15);
    WordDocument1.Tables.Item(1).Columns.Item(5).Width:=Round((W-5*k)*0.5);

  if Button2.Tag = 0 then WordDocument1.PrintOut;
  WordEnd(FileName);
end;

procedure TFMain.E_FindNumClick(Sender: TObject);
begin
  RB1.Checked:=True;
end;

procedure TFMain.E_FindTextClick(Sender: TObject);
begin
  RB2.Checked:=True;
end;

procedure TFMain.ME_FindDateClick(Sender: TObject);
begin
```

```
  RB3.Checked:=True;
end;

{============EMAIL===============}
procedure TFMain.Button20Click(Sender: TObject);
  var InputString: string;
begin
  if IdSMTP1.Connected=True then
  begin
    InputString:= InputBox('Êóäà îòïðàâèòü', 'E-mail', '');
                                                    Msg.Subject:='Ïðèêàç';
Msg.Recipients.EMailAddresses:=InputString;
Msg.From.Address:= Login + '@' + Addr;
Msg.Body.Text   := ME_DateDecree.Text + '   ' +
                   E_NumDecree.Text + '    ' +
                   M_DecreeText.Text;
Msg.Date:=date;
Msg.ContentType:='text/html; charset=windows-1251';
Msg.ContentTransferEncoding:='8bit';

   if idSmtp1.Connected = TRUE
   then IdSMTP1.Send(Msg);

   Application.ProcessMessages;
   idSmtp1.Disconnect;
  end;
end;

procedure TFMain.Button2Click(Sender: TObject);
var s :string;
InputString: string;
begin
  if IdSMTP1.Connected=True then
  begin
    Button2.Tag:=1;
    InputString:= InputBox('Êóäà îòïðàâèòü', 'E-mail', '');
    Msg.Clear;
    Msg.Subject:='Îò÷åò';
    Msg.Recipients.EMailAddresses:=InputString;
    Msg.From.Address:=Login+'@'+Addr;
    Msg.Body.Text:='Îò÷åò';
    Msg.Date:=date;
    case SGZ.Tag of
      2:begin print2; s:=PathProg + '\report\rep1.tmp'; end;
      3:begin print3; s:=PathProg + '\report\rep2.tmp'; end;
      4:begin print4; s:=PathProg + '\report\rep1.tmp'; end;
      5:begin print5; s:=PathProg + '\report\rep3.tmp'; end;
      6:begin print6; s:=PathProg + '\report\rep3.tmp'; end;
      7:begin print7; s:=PathProg + '\report\rep3.tmp'; end;
    end;

     CopyFile(pchar(S), pchar(PathProg + '\report\sendrep' + IntToStr(SGZ.Tag) +
'.doc'), true);
    s:=PathProg + '\report\sendrep' + IntToStr(SGZ.Tag) + '.doc';
    //s:=  GetCurrentDir+'\report\qqq.doc';

    TIdAttachment.Create(Msg.MessageParts,s);
    IdSMTP1.Send(Msg);
    //IdSMTP1.Disconnect;
  end;
  BUtton2.Tag:=0;
end;
```

```
procedure TFMain.Connect1Click(Sender: TObject);
begin
  Cursor :=  crHourGlass;
  IdSMTP1.Host := SMTPSer;
  //IdSMTP1.AuthenticationType:=atLogin;
  IdSMTP1.UserId:=Login + '@' + Addr;
  IdSMTP1.Password:=Pass;
  IdSMTP1.Authenticate;
  //showmessage(IdSMTP1.UserId);
  try
    IdSMTP1.Connect;
  except
    ShowMessage('Îøèáêà ïðè ïîäêëþ÷åíèè ê ñåðâåðó!');
    Cursor := crDefault;
  end;
  ShowMessage('Ïðîèçâåäåíî ïîäêëþ÷åíèè ê ïî÷òîâîìó ñåðâåðó!');
end;

procedure TFMain.IdSMTP1Connected(Sender: TObject);
begin
  Button2.Enabled  := True;
  Button20.Enabled := True;
  Cursor := crDefault;
end;

procedure TFMain.IdSMTP1Disconnected(Sender: TObject);
begin
  Button2.Enabled  := False;
  Button20.Enabled := False;
end;

procedure TFMain.Button25Click(Sender: TObject);
begin
 FExecutor.ShowModal;
end;

procedure TFMain.B_EditCaseClick(Sender: TObject);
begin
  if DM.IBQ_Cases.IsEmpty
  then exit;

   if (DBLkpCB_Dep2.KeyValue=-1) or (E_NumCase.Text='') or (ME_DateCase.Text='')
or
     (CB_IndexCase.Text='') or (M_InfoCase.Text='')
  then begin
       showmessage('Íåîáõîäèìû ââîä äàííûõ!');
       exit;
       end;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
       SQL.Add(format('UPDATE  CASES  SET  DEP=%s,  CASE_NUM=''%s'',  IND=''%s'',
CASE_DATE=''%s'', CASE_TEXT=''%s'' WHERE ID=%s',
              [
               IntToStr(DBLkpCB_Dep2.KeyValue),
               E_NumCase.Text,
               CB_IndexCase.Text,
               ME_DateCase.Text,
               M_InfoCase.Text,
               DM.IBQ_CasesID.asString
```

```
            ]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Cases.Close;
    IBQ_Cases.Open;
  end;
end;

procedure TFMain.Disconnect1Click(Sender: TObject);
begin
 IdSMTP1.Disconnect;
end;

procedure TFMain.IdSMTP1WorkEnd(Sender: TObject; AWorkMode: TWorkMode);
begin
  //IdSMTP1.Disconnect;
end;

procedure TFMain.DBG_DepDblClick(Sender: TObject);
begin
  if DM.IBQ_Dep.IsEmpty
  then exit;

  E_DepName.Text:=DM.IBQ_DepDEP.AsString;
  M_DepInfo.Text:=DM.IBQ_DepINFO.AsString;
end;

procedure TFMain.DBG_TypeCasesDblClick(Sender: TObject);
begin
  with DM do
  begin
    if IBQ_TypeCases.IsEmpty
    then exit;

    DBLkpCB_Dep.KeyValue        :=IBQ_TypeCasesDEP.AsInteger;
    E_TypeCase_Index.Text       :=IBQ_TypeCasesIND.AsString;
    SE_TypeCase_CountPart.Value :=IBQ_TypeCasesCOUNTPART.AsInteger;
    E_TypeCase_Period.Text      :=IBQ_TypeCasesPERIOD.AsString;
    E_TypeCase_Num.Text         :=IBQ_TypeCasesPERNUM.AsString;
    M_TypeCaseName.Text         :=IBQ_TypeCasesNAMECASE.AsString;
    M_TypeCaseName.Text         :=IBQ_TypeCasesNAMECASE.AsString;
    E_Info.Text                 :=IBQ_TypeCasesINFO.AsString;

  end;
end;

procedure TFMain.DBG_DecreesDblClick(Sender: TObject);
begin
  with DM do
  begin
    if IBQ_Decrees.IsEmpty
    then exit;

    E_NumDecree.Text       :=IBQ_DecreesDECREE_NUM.AsString;
    ME_DateDecree.Text     :=IBQ_DecreesDATE_.AsString;
    M_DecreeText.Text      :=IBQ_DecreesDECREE_TEXT.AsString;
```

```
  end;
end;

procedure TFMain.DBG_CasesDblClick(Sender: TObject);
begin
  with DM do
  begin
    if IBQ_Cases.IsEmpty
    then exit;

    DBLkpCB_Dep2.KeyValue :=IBQ_CasesDEP.AsInteger;
    DBLkpCB_Dep2Exit(Sender);
    E_NumCase.Text        :=IBQ_CasesCASE_NUM.AsString;
    CB_IndexCase.Text     :=IBQ_CasesIND.AsString;
    ME_DateCase.Text      :=IBQ_CasesCASE_DATE.AsString;
    M_InfoCase.Text       :=IBQ_CasesCASE_TEXT.AsString;
  end;
end;

procedure TFMain.ME_DateDecreeChange(Sender: TObject);
  var S: string;
      ME:TMaskEdit;
begin
  ME := Sender as TMaskEdit;
  ME.Font.Color := clWindowText;
  S:=ME.Text;

  if (S[1]<>' ') or (S[2]<>' ') or (S[4]<>' ') or (S[5]<>' ') or
     (S[7]<>' ') or (S[8]<>' ') or (S[9]<>' ') or (S[10]<>' ')
  then begin
       if (S[1]<>' ') and (S[2]<>' ') and (StrToInt(S[1]+S[2])>31)
       then ME.Font.Color := clRed;

        if (S[4]<>'  ') and (S[5]<>'  ') and ((StrToInt(S[4] + S[5])>12) or
(StrToInt(S[4] + S[5])<1))
       then ME.Font.Color := clRed;

       if (S[7]=' ') or (S[8]=' ') or (S[9]=' ') or (S[10]=' ')
       then ME.Font.Color := clRed
        else if (StrToInt(S[7]+S[8] + S[9] + S[10])>2025) or (StrToInt(S[7] +
S[8] + S[9] + S[10])<1990)
            then  ME.Font.Color := clRed;

       if (s[1] + s[2]='  ') and (s[4] + s[5] <> '  ')
       then ME.Font.Color := clRed;

       if (s[4] + s[5]='  ') and (s[7] + s[8] + s[9] + s[10]<>'    ')
       then ME.Font.Color := clRed;
       end;
end;

procedure TFMain.DBLkpCB_Dep2Exit(Sender: TObject);
  var s:string;
      i:integer;
begin
  if DBLkpCB_Dep2.Text<>'' then
  begin
    CB_IndexCase.Clear;

            s:='SELECT   IND,   NAMECASE   FROM   TYPECASES   WHERE   DEP='   +
IntToStr(DBLkpCB_Dep2.KeyValue);
```

```
    with DM.IBQuery1 do
    begin
      SQL.Clear;
      SQL.Add('COMMIT;');
      ExecSQL;

      SQL.Clear;
      SQL.Add(s);
      Open;

      First;
      i:=0;
      while not EOF do
      begin
      CB_IndexCase.Items.Add(Fields[0].AsString);
      tn3[i]:=Fields[1].AsString;
      inc(i);
      Next;
      end;
      CB_IndexCase.ItemIndex :=0;
      SQL.Clear;
      SQL.Add('COMMIT;');
      ExecSQL;
    end;
  end;
end;

procedure TFMain.B_EditInCorClick(Sender: TObject);
begin
  if DM.IBQ_InCors.IsEmpty
  then exit;

  if (ME_InCorDate1.Text='') or (E_Sender.Text='') or
     (E_InCorFhN.Text='') or (ME_InCorDate2.Text='') or
     (E_InCorSign.Text='') or (M_InCorText.Text='')  or
     (DBLkpCB_Executors.KeyValue=-1)
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
    SQL.Add(format('UPDATE INCORS SET DATE1=''%s'', SENDER=''%s'', NUMVH=''%s'',
DATE2=''%s'', INCOR_TEXT=''%s'', ID_EXECUTOR=%s, INFO=''%s'' WHERE ID=%s',
               [
                ME_InCorDate1.Text,
                E_Sender.Text,
                E_InCorFhN.Text,
                ME_InCorDate2.Text,
                M_InCorText.Text,
                IntToStr(DBLkpCB_Executors.KeyValue),
                '',
                DM.IBQ_InCorsID.asString
               ]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
```

```
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_InCors.Close;
    IBQ_InCors.Open;
  end;

  {ME_InCorDate1.Text:='';
  E_Sender.Text:='';
  E_InCorFhN.Text:='';
  ME_InCorDate2.Text:='';
  M_InCorText.Text:='';
  DBLkpCB_Executors.KeyValue:=-1;}
end;

procedure TFMain.B_NewOutCorClick(Sender: TObject);
begin
  if (ME_OutCorDate.Text='') or (E_SendTo.Text='') or
     (M_OutCorText.Text='') or (DBLkpCB_Dep3.KeyValue=-1)
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
    SQL.Add('INSERT INTO OUTCORS(DATE1, SENDTO, OUTCOR_TEXT, DEP)');
    SQL.Add(format('VALUES(''%s'', ''%s'', ''%s'', %s)',
               [ME_OutCorDate.Text,
                E_SendTo.Text,
                M_OutCorText.Text,
                IntToStr(DBLkpCB_Dep3.KeyValue)
               ]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_OutCors.Close;
    IBQ_OutCors.Open;
  end;
  ME_OutCorDate.Text:='';
  E_SendTo.Text:='';
  M_OutCorText.Text:='';
  DBLkpCB_Dep3.KeyValue:=-1;
end;

procedure TFMain.DBG_InCorsDblClick(Sender: TObject);
begin
  with DM do
  begin
    if IBQ_InCors.IsEmpty
    then exit;
    ME_InCorDate1.Text         :=IBQ_InCorsDATE1.AsString;
    E_Sender.Text              :=IBQ_InCorsSENDER.AsString;
    E_InCorFhN.Text            :=IBQ_InCorsNUMVH.AsString;
    ME_InCorDate2.Text         :=IBQ_InCorsDATE2.AsString;
    M_InCorText.Text           :=IBQ_InCorsINCOR_TEXT.AsString;
```

```
    DBLkpCB_Executors.KeyValue :=IBQ_InCorsID_EXECUTOR.AsInteger;
  end;
end;

procedure TFMain.B_EditOutCorClick(Sender: TObject);
begin
  if DM.IBQ_OutCors.IsEmpty
  then exit;

  if (ME_OutCorDate.Text='') or (E_SendTo.Text='') or
     (M_OutCorText.Text='') or (DBLkpCB_Dep3.KeyValue=-1)
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;
  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
          SQL.Add(format('UPDATE   OUTCORS   SET   DATE1=''%s'',   SENDTO=''%s'',
OUTCOR_TEXT=''%s'', DEP=%s WHERE ID=%s',
               [
                 ME_OutCorDate.Text,
                 E_SendTo.Text,
                 M_OutCorText.Text,
                 IntToStr(DBLkpCB_Dep3.KeyValue),
                 DM.IBQ_OutCorsID.asString
               ]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_OutCors.Close;
    IBQ_OutCors.Open;
  end;

end;

procedure TFMain.B_EditDecreeClick(Sender: TObject);
begin
  if DM.IBQ_Decrees.IsEmpty
  then exit;
  if (E_NumDecree.Text='') or (M_DecreeText.Text='')
  then begin
       showmessage('Íåïîëíûé ââîä äàííûõ!');
       exit;
       end;
  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
     SQL.Add(format('Update DECREES Set DECREE_NUM=''%s'', DECREE_TEXT=''%s'',
DATE_=''%s'' Where ID=%s',
               [
                 E_NumDecree.Text,
                 M_DecreeText.Text,
                 ME_DateDecree.Text,
                 DM.IBQ_DecreesID.asString
               ]));
    try
```

```
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Decrees.Close;
    IBQ_Decrees.Open;
  end;
end;

procedure TFMain.B_DelDecreeClick(Sender: TObject);
begin
  if DM.IBQ_Decrees.IsEmpty
  then exit;
  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
                SQL.Add(format('Delete    from    DECREES    Where    ID=%s',
[DM.IBQ_DecreesID.asString]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Decrees.Close;
    IBQ_Decrees.Open;
  end;
  E_NumDecree.Text:='';
  M_DecreeText.Text:='';
  ME_DateDecree.Text:='';
end;

procedure TFMain.B_DelCaseClick(Sender: TObject);
begin
  if DM.IBQ_Cases.IsEmpty
  then exit;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
    SQL.Add(format('Delete from CASES Where ID=%s', [DM.IBQ_CasesID.asString]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_Cases.Close;
    IBQ_Cases.Open;
  end;
  DBLkpCB_Dep2.KeyValue:=-1;
  E_NumCase.Text:='';
  CB_IndexCase.Text:='';
  ME_DateCase.Text:='';
  M_InfoCase.Text:='';
```

```
end;

procedure TFMain.B_DelInCorClick(Sender: TObject);
begin
  if DM.IBQ_InCors.IsEmpty
  then exit;

  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
                  SQL.Add(format('Delete    from    INCORS    Where    ID=%s',
[DM.IBQ_InCorsID.asString]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_InCors.Close;
    IBQ_InCors.Open;
  end;

  ME_InCorDate1.Text:='';
  E_Sender.Text:='';
  E_InCorFhN.Text:='';
  ME_InCorDate2.Text:='';
  M_InCorText.Text:='';
  DBLkpCB_Executors.KeyValue:=-1;
end;

procedure TFMain.B_DelOutCorClick(Sender: TObject);
begin
  if DM.IBQ_OutCors.IsEmpty
  then exit;
  with DM, DM.IBQuery1 do
  begin
    SQL.Clear;
                 SQL.Add(format('Delete    from    OUTCORS    Where    ID=%s',
[DM.IBQ_OutCorsID.asString]));
    try
      ExecSQL;
    except
      Transaction.RollbackRetaining;
      ShowMessage('Îøèáêà âûïîëíåíèÿ!');
      exit;
    end;
    Transaction.CommitRetaining;
    IBQ_OutCors.Close;
    IBQ_OutCors.Open;
  end;

  ME_OutCorDate.Text:='';
  E_SendTo.Text:='';
  M_OutCorText.Text:='';
  DBLkpCB_Dep3.KeyValue:=-1;
end;

end.
```