ОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Темой настоящего дипломного проекта является разработка автоматизированной системы управления персоналом.

Отдел кадров или его аналог всегда существует на любом предприятии, в любой организации. Даже если размеры организации невелики, функции отдела кадров на себя берет (по совместительству) кто-нибудь из сотрудников. Этим человеком может быть даже индивидуальный предприниматель, имеющий право выступать в качестве работодателя. Прием на работу, увольнение сотрудника, предоставление ему отпуска, отгула и многие другие действия должны быть правильно оформлены с точки зрения действующего законодательства. Конечно, в настоящее время такая работа невозможна без применения соответствующего программного обеспечения, как минимум потому, что бланки отчетов стандартизированы и принимаются только в электронном виде. Есть множество других причин, без которых работа отдела кадров становится невозможной без использования программных средств: в частности, необходимо иметь список работников, возможность их сортировки и фильтрации, необходимо формирование электронных документов для документооборота организации и т.п.

На рынке всегда есть большое количество различных программных продуктов, призванных помочь быстро и качественно решить кадровые вопросы. Их сравнительный анализ будет произведен в ходе исследования. Пока же достаточно сказать о том, что их ассортимент очень разнообразен – у пользователя есть возможность выбирать программы начиная с самых простейших (и соответственно, бесплатных), не требующих много времени на освоение, до сложных автоматизированных информационных систем предприятия, для приобретения которых требуются значительные финансовые средства, а для работы в их среде – обучение персонала.

Цель дипломной работы – исследование возможности автоматизации работы отдела кадров без создания базы данных с практической реализацией в виде разработки программного обеспечения с заданными параметрами, имеющего практическую направленность.

Задачи:

1) Анализ объекта информатизации
2) Анализ предложений на рынке специализированного ПО для отдела кадров
3) Обоснование выбора программных средств решения задачи
4) Разработка алгоритма решения задачи
5) Разработка форм входных/выходных данных и оптимального дизайна

6) Реализация поставленных целей с помощью функциональных возможностей выбранной среды разработки.

Актуальность работы заключается в возможности разработки бесплатного приложения, способного взять на себя функции программных комплексов для управления кадрами небольшой организации.

Объект исследования: функции отдела кадров предприятия, программные продукты соответствующей направленности, документационное обеспечение.

Предмет исследования – функциональные возможности выбранной среды разработки.

Методы исследования: изучение литературы, сравнительный анализ, моделирование.

Новизна разработки: создание программы, имеющей функции базы данных, без использования соответствующих компонентов.

Необходимо сразу же сделать оговорку, что в данной работе не ставится цель создать программный продукт, обладающий полноценным функционалом и способный заменить собой коммерческие разработки крупных производителей софта. Полученный продукт должен стать лишь небольшим подспорьем для сотрудника отдела кадров или лица, его заменяющего, в тех случаях, когда использование более сложного ПО в силу каких-то причин затруднительно. Его основные достоинства – простота интерфейса, быстрота работы, отсутствие ошибок и защита от несанкционированного использования. Функции программного продукта более полно будут раскрыты в дальнейшем описании.

# 1. ОБЩАЯ ЧАСТЬ

## 1.1. Характеристика объекта, процесса, предметной области как объекта информатизации

Система кадровой документации достаточно специфична. В работе с личным составом создаётся большой объём однотипных документов, легко поддающихся формализации.

Создание унифицированных форм этих документов и их электронных версий, использование специальных компьютерных программ значительно облегчает работу всех, кто работает с кадровыми документами. Применение компьютера делает возможным даже силами одного специалиста выполняющего функции кадровика, вести документацию по личному составу средних и даже крупных организаций.

Служба кадров сегодня оснащена современной оргтехникой, специализированными программами, обеспечивающими однократный ввод информации по личному составу и её использование всеми подразделениями.

Внедрение компьютерной техники позволяет накапливать массивы информации (базы данных) и документы в электронной форме по всем сотрудникам организации, кадровому резерву, быстро находить и эффективно обрабатывать всю необходимую информацию по личному составу.

Кадровая документация - непременная часть документов любой организации. Она ведётся кадровой службой (отделом, департаментом по работе с персоналом).

Использование автоматизированных технологий позволяет существенно повысить эффективность работы кадровой службы. Документы по личному составу - это специфическая документация, требующая специализированных программ для работы с ней. Сегодня практически в любой организации есть своя компьютерная служба. В небольшом офисе - это один или несколько «компьютерщиков». В средних и крупных организациях - это отделы, департаменты информационного обеспечения (так называемые информационные (IT) службы). Однако работники ITслужб - это программисты, системные администраторы, то есть специалисты именно по компьютерной технике. А для составления списка требований к системе автоматизации службы кадров, выбора в наибольшей степени подходящей для неё системы программного обеспечения (ПО) необходимы усилия именно специалистов самой кадровой службы. Поэтому работники службы кадров должны хорошо ориентироваться в возможностях современных автоматизированных систем, в первую очередь - для определения потребностей своего подразделения и составлении оптимального задания для IT-служб на приобретение,

установку, настройку и последующее обслуживание средств автоматизации, необходимых в работе с документацией по личному составу.

Рынок информационных технологий предлагает широкий выбор программных продуктов для решения актуальных проблем управления персоналом и оптимизации бизнес-процессов в компаниях разного уровня организационного развития и разных направлений деятельности.

Автоматизированная система управления кадровой службой позволяет вести учет работников, издавать и регистрировать приказы, следить за предоставлением отпусков, получать разнообразную аналитическую информацию и решать многие другие задачи. Программы управления человеческими ресурсами помогают не только эффективно распределять трудовые ресурсы и управлять капиталом, но и являются источником, из которого служащие могут получать сведения как корпоративного, так и индивидуального характера.

Системы учета кадров были разработаны на основании программ расчета заработной платы. В дальнейшем функционал этих программ значительно расширился. Это было связано с осознанием руководителей предприятий необходимости качественных изменений в работе отдела кадров. Оптимизация работы персонала, рост профессионализма специалистов по работе с персоналом приводит к тому, что деятельность по управлению кадрами становится все более технологичной, системной и качественной. Успех любой компании зависит часто от того, насколько эффективным будет работа отдела кадров, задача которого - найти и удержать нужных специалистов, а также правильно распределить имеющиеся трудовые ресурсы. Иногда при этом приходится вносить коррективы в структуру компании. Соответственно растут и потребности в эффективных инструментах управления потоками информации. В связи с этим разработчики автоматизированных систем должны уделять больше внимания развитию программных продуктов по управлению человеческими ресурсами, искать пути объединения необходимых функций в единую информационную систему управления кадрами.

Программный продукт по управлению кадровой службой позволяет:

- оперативно получать аналитическую информацию и принимать обоснованные управленческие решения;
- организовать бизнес-процессы по управлению персоналом, исключить многократный ввод одних и тех же данных в учетную систему и оптимизировать ежедневную работу сотрудников различных служб компании;
- наладить эффективный учет всей информации, относящейся к персоналу компании, создавая тем самым основу для анализа и планирования затрат на персонал;

- вести учет в соответствии с законодательством и минимизировать риск финансовых санкций со стороны фискальных органов.

На сегодняшний день существует достаточно много систем управления кадровой службой, представленных как комплексными программами, которые охватывают весь диапазон задач управления человеческими ресурсами, так и узкоспециализированными решениями.

### 1.2. Анализ предметной области

Как и в любом проекте, начинать необходимо с определения главной и промежуточных целей проекта. Именно от правильно расставленных на этом этапе акцентов будет во многом зависеть успех проекта автоматизации, то есть выбирать продукт для автоматизации следует в соответствии с поставленными задачами.

Многие пытаются пойти обратным путем - сначала выбрать информационную систему управления кадровой службы, а затем с ее помощью решить определенные задачи. Такой шаг заведомо неверный, так как все существующие на отечественном рынке программы имеют различный функционал и не могут претендовать на звание «универсальной системы».

Итак, в первую очередь необходимо определить: для чего нужна автоматизация, какие требования предъявить к информационной системе и каких результатов от нее ожидать.

Важно автоматизировать не функции управления персоналом вообще, а именно те из них, которые важны на текущий момент и будут востребованы в ближайшее время. Избыточные функции системы затруднят работу пользователей и отнимут дополнительные ресурсы «компьютерного парка» компании.

Обычно среди автоматизируемых функций - основные и наиболее трудоемкие бизнес-процессы, связанные с управлением персоналом: прием на работу, перевод, увольнение, оформление отпусков и т. д. Также в автоматизации нуждаются такие процессы, как учет рабочего времени и начисление заработной платы. Компаниям, бизнес-процессы которых ориентированы на западные стандарты, необходима автоматизация планирования карьеры, управления обучением, подбора кандидатов, планирования организационной структуры и штатного расписания, самообслуживания. Такой функциональностью интересуются в первую очередь быстро развивающиеся компании либо предприятия с западным стилем менеджмента (в основном - представительства иностранных компаний).

Основные потребности большинства отечественных предприятий определяются двумя факторами: общей ситуацией с автоматизацией и требованиями законодательства. На сегодняшний день автоматизация учета кадров на многих предприятиях - «кусочно-лоскутная»: приказы печатаются в «Word», кадровый учет ведется в «Excel», зарплата считается в «1С». При большой численности работников трудозатраты на получение статистических данных, а также на проверку информации, получаемой руководством из разных подразделений, непомерно возрастают. Поэтому первостепенная задача - навести элементарный порядок в кадровом учете.

Необходимо отметить, что, несмотря на растущий спрос, автоматизация работы кадровой службы не является наиболее приоритетной задачей автоматизации в большинстве компаний. В первую очередь автоматизируются бухгалтерский, налоговый, производственный и оперативный учет. Поэтому бюджеты проектов по автоматизации управления кадрами существенно меньше бюджетов на автоматизацию, скажем, бухгалтерского учета, хотя по трудозатратам эти задачи вполне сравнимы. Довольно часто по этой причине на первом этапе автоматизации приходится ограничиваться решением наиболее важных и сложных задач.

Использование новых технологий значительно повышает эффективность работы кадровых служб за счет использования преимуществ информационных систем. Важно не ошибиться с определением как самого программного обеспечения, так и компании-поставщика. В случае неудачного выбора могут не оправдаться предварительные ожидания, и, как следствие, - «ручная работа», удорожание проекта внедрения, использование системы с ущербом для производительности (с учетом уже затраченных временных и финансовых средств на ее приобретение и внедрение), а в итоге, возможно, полный отказ от программного обеспечения.

При выборе программного обеспечения следует выделить несколько ключевых этапов, определяющих успешность проекта автоматизации в целом. В первую очередь необходимо руководствоваться сравнением начальных функциональных возможностей систем (типовой конфигурации) с поставленными задачами. На этом этапе также необходимо оценить гибкость продукта, т. е. степень легкости его адаптации к специфическим требования будущих пользователей.

Вторым важнейшим этапом является приспособление продукта к требованиям национального законодательства. Повышенным спросом пользуются системы с оперативной поддержкой изменений в соответствии с требованиями законодательства, опробованные в реальной работе на российских предприятиях.

Следующим этапом является соотношение цены и полезного функционала сравниваемых систем. При этом необходимо учитывать такие показатели, как простота использования, быстродействие, необходимые технические средства, надежность и пр. Очень важно обратить внимание не только на цену программного продукта, но и на совокупную стоимость программного обеспечения, консультационных услуг и послепроектного сопровождения системы. Иногда стоимость внедрения может ощутимо превышать стоимость программы.

Информационные системы - это интеллектуальные продукты, поэтому успех автоматизации часто зависит в большей степени от команды специалистов по внедрению и поддержке информационного продукта.

Качество внедрения, сроки выполнения проекта, решение поставленных задач и общая удовлетворенность результатами проекта напрямую зависят от опыта и знаний консультантов компании-поставщика.

При выборе компании-внедренца следует учесть количество выполненных проектов предлагаемого программного обеспечения, обратить внимание на стабильность, опыт и репутацию этой компании на рынке и т. д.

Не стоит забывать, что сведения о неудачных проектах быстро распространяются. И если ваша компания ориентирована на долгосрочные перспективы в бизнесе, то выбирать партнера в сфере автоматизации надо на основании его успешного опыта работы на отечественном рынке.

Прежде всего, руководитель предприятия должен четко осознавать, что эффективность бизнеса повышают не только компьютерные программы, но и правильные решения и действия персонала компании. При этом роль систем автоматизации заключается в своевременном предоставлении максимально полной информации для принятия соответствующих решений.

Оптимальным путем внедрения проекта автоматизации обычно является максимальное привлечение к работам по проекту сотрудников предприятия-заказчика, использующих советы и опыт специалистов компании-внедренца.

Автоматизация любого процесса - серьезный проект, требующий постоянного внимания и участия будущих пользователей. В силу слабой формализации процессов кадрового менеджмента управленческие модули информационных систем управления кадрами не диктуют жестких моделей и схем. Автоматизация отдела кадров - процесс не только технический, но и творческий. К сожалению, в разработке информационных систем профессиональные кадровики принимают недостаточное участие, а заложенные в системы «книжные» модели управления кадрами не всегда пригодны для реальных управленческих

процессов. Профессионализм и практический опыт специалистов-кадровиков, четкое понимание задачи и творческий подход к делу будут очень уместны для развития корпоративной системы управления кадрами.

Очень важным фактором является сопровождение системы. Пакеты управления кадрами менее подвержены изменениям, чем модули расчета зарплаты, но, тем не менее, должны также своевременно и качественно сопровождаться. Этот вопрос должен подробно обсуждаться при переговорах с потенциальными подрядчиками работ по внедрению программы.

Стоит отметить, что в последнее время на отечественном рынке систем автоматизации кадрового учета:

- вырос спрос на «западную» функциональность;
- увеличился интерес к Internet-технологиям;
- наблюдается интеграция с корпоративными ERP-системами и со смежными продуктами;
- развиваются партнерские сети, а также методологические базы проектов внедрения и консультирования.

В ближайшее время прогнозируется повышение спроса на комплексные системы управления кадрами, регистрации рабочего времени и расчета зарплаты, соответствующие законодательству.

### 1.3. Программное обеспечение для работы отдела кадров

Прежде чем начать разработку собственного программного обеспечения любой предметной области, необходимо посмотреть, какие принципы положены в основу, какие требования учитываются прежде всего, и это невозможно сделать без знакомства с ПО различных производителей.

Уже первое знакомство показывает, что все важные объекты сгруппированы по определенному признаку – без этого невозможно создание ни одной базы данных, независимо от того, какая среда разработки используется. Чем сложнее и дороже программный комплекс, тем таких групп объектов больше, тем изощреннее и сложнее связи между ними. Разумеется, цена растет прямо пропорционально сложности.

Одним из самых популярных программных комплексов для решения кадровых задач является платформа 1С:Предприятие с конфигурацией «Зарплата и управление персоналом».

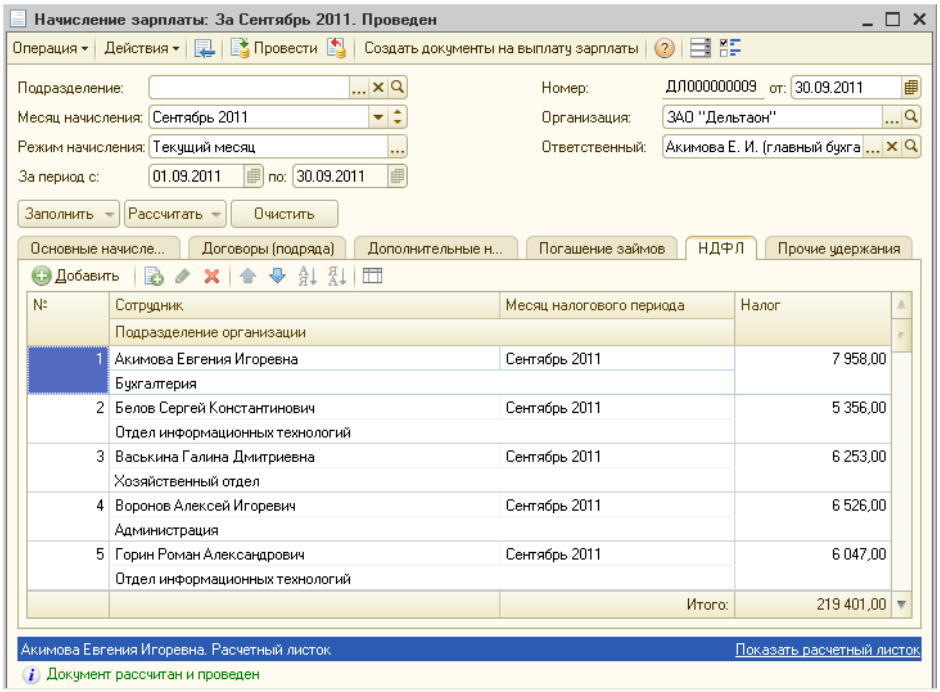



Рис . 1.1. Скриншот документа «Начисление зарплаты» в среде 1С: Зарплата и Управление Персоналом»

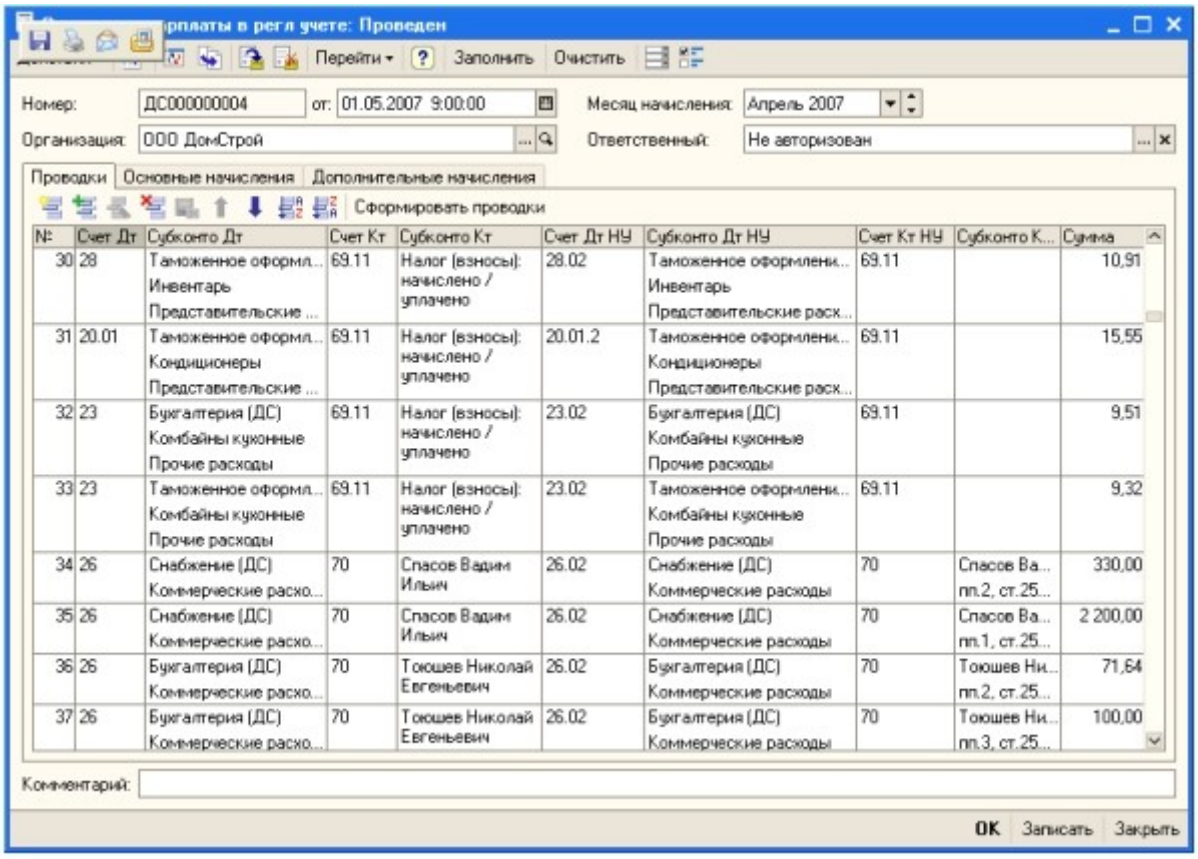
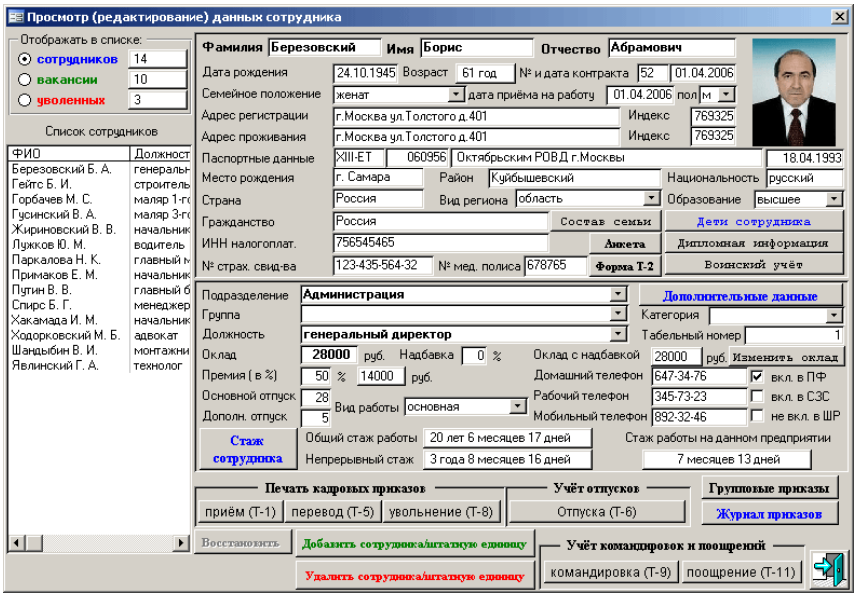


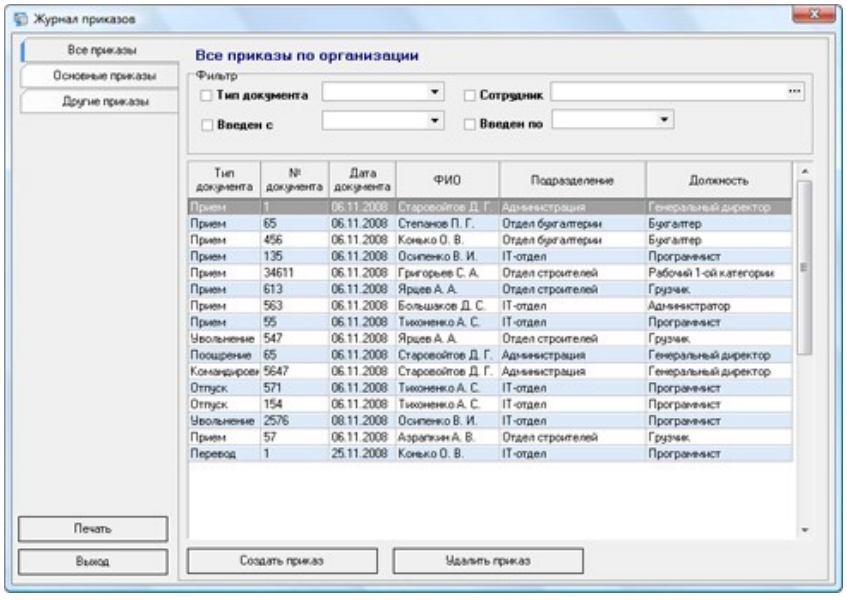

Рисунок 1.2. Окно документа в среде 1С: Зарплата и Управление Персоналом»

Безусловно, 1С является одной из лучших программ в своём классе. Ее преимущества можно перечислять долго: начиная от огромного количества документов и позиций учета до высокой, почти абсолютной наадежности расчетов и сохранности данных. Конфигурации 1С достаточно легко адаптировать под конкретные требования. Кроме этого, фирма 1С очень трепетно относится к своим клиентам, постоянно выпускает обновления, проводит различные семинары, конференции, оказывает клиентам всемерную поддержку.

Однако и у конфигураций 1С есть недостатки: рабочая среда не так проста для освоения, интерфейс зачастую перегружен деталями, не самая маленькая цена, и это еще далеко не всё. Поэтому нельзя сказать, что конфигурации 1С являются стандартами де-факто в Российской Федерации.

Одним из конкурентов фирмы 1С в области кадрового учета и управления персоналом является фирма «Новая высота» из Новосибирска, выпускающая программу «Отдел кадров Плюс». Ниже приведены несколько скринштов программы.



Также на рынке есть ПО разработки ООО «Бравософт» под названием «Персонал».

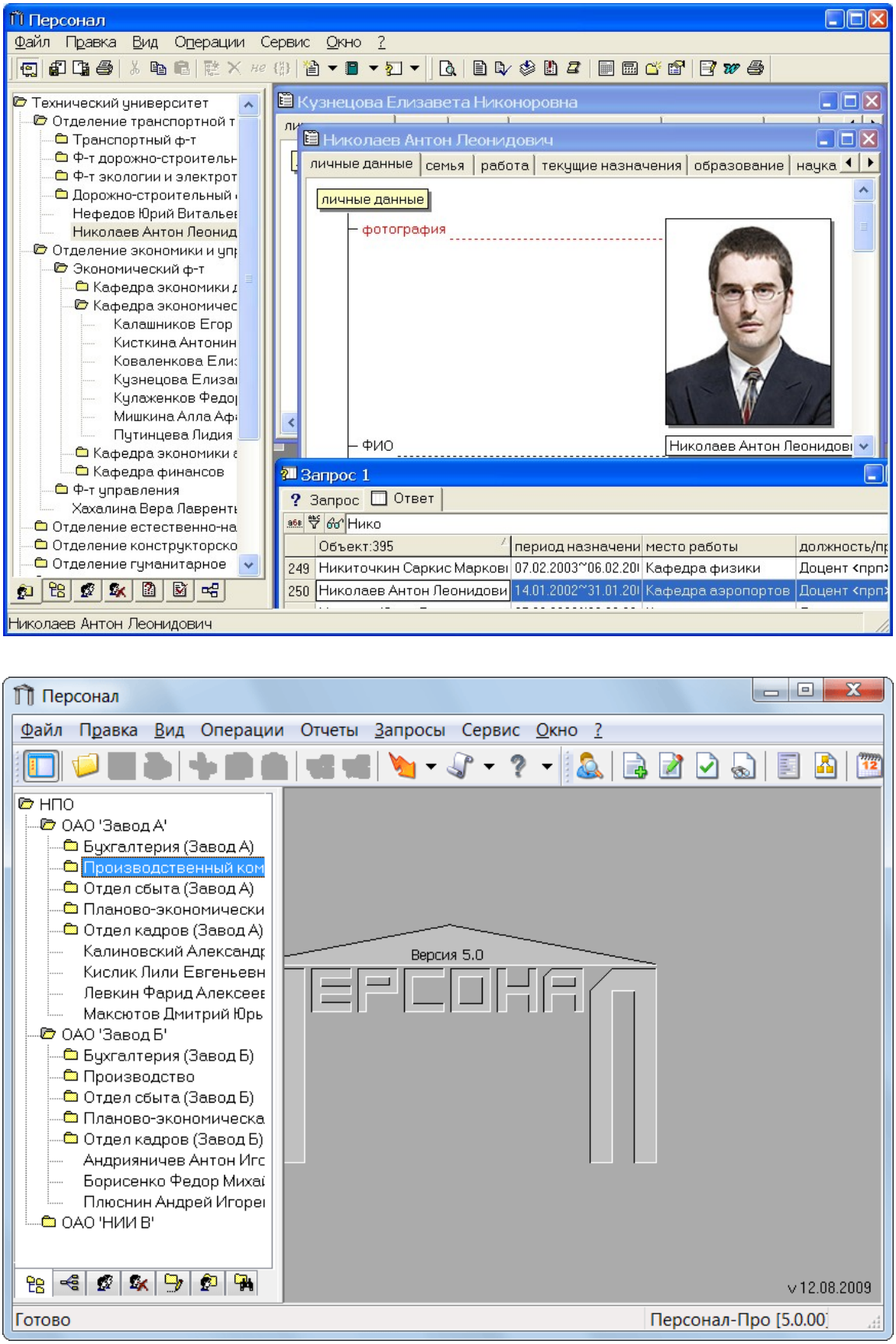






Рисунок 1.5. Программа «Персонал» от ООО «Бравософт»

На самом деле, программ для учета персонала существует очень большое количество, их производят как крупные компании, так и маленькие, а число программистов-энтузиастов, разрабатывающие и выкладывающие в интернет бесплатные приложения, и вовсе не поддается учету. В данном проекте нет возможности, да и необходимости, рассмотреть хотя бы малую часть из них, важно было понять, что является приоритетным при разработке таких программ.

### 1.4. Минимальные системные требования для работы программы

Таблица 1

| Параметр | Значение | Размерность |
|---|---|---|
| Емкость жёсткого диска, не менее | 40 | Gb |
| Частота процессора, не менее | 1,3 | Hhz |
| Объем оперативной памяти, не менее | 1 | Gb |
| Операционная система | Windows 7, 8, 10 | |
| Обязательное наличие дополнительной видеокарты | нет | |

# 2. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## 2.1. Постановка задачи

Разработать программный продукт со следующими характеристиками:

- Простой интерфейс;
- Минимальное количество окон;
- Модальное отображение подчиненных окон;
- Наличие главной формы приложения;
- Невозможность прямого редактирования элементов списка;
- Защита данных от несанкционированного доступа;
- Возможность регистрации;
- Автоматическое структурирование файлов списков.

## 2.2. Обоснование выбора программных средств решения задачи

Средой разработки выбран программный комплекс Embarcadero RAD Studio XE8 Architect.

C++Builder и Delphi стали одними из самых популярных на сегодняшний день инструментов для создания как настольных, так и корпоративных информационных систем благодаря уникальному сочетанию удобства разработки пользовательских интерфейсов, компонентной архитектуры, однотипности доступа к разнообразным базам данных, начиная от плоских таблиц формата dBase и Paradox и кончая серверными СУБД. Во многом именно наличие таких продуктов стимулировало достаточно безболезненный перенос в архитектуру клиент/сервер ряда информационных систем, модернизация которых иными средствами была бы сопряжена с большими трудовыми и материальными затратами.

Следует отметить, что современные тенденции развития инструментальных средств таковы, что актуальным становится не просто появление новых гибких и мощных средств разработки, а создание семейств таких продуктов с похожими средами и принципами создания приложений, что в це- лом повторяет появившуюся примерно 4 года назад идеологию формирования офисных пакетов (текстовый процессор + электронная таблица + настольная СУБД + презентационный пакет) вместо выпуска отдельных офисных приложений. Если рассматривать линию продуктов Inprise, то в данный момент на рынке

средств разработки присутствуют Delphi и C++Builder, а также сходные по методам создания приложений и среде JBuilder, IntraBuilder, Visual dBase.

Сходство C++Builder и Delphi не является чисто внешним. C++Builder обладает компонентной архитектурой и создан на основе библиотеки визуальных компонентов Delphi ставшей за последние два года весьма популярной среди разработчиков. По этой причине этот продукт имеет общую с Delphi библиотеку классов, часть из которых написана на Obiect Pascal.

Сходство C++Builder и Delphi не является чисто внешним. C++Builder обладает компонентной архитектурой и создан на основе библиотеки визуальных компонентов Delphi, ставшей за последние два года весьма популярной среди разработчиков. По этой причине этот продукт имеет общую с Delphi библиотеку классов, часть из которых написана на Object Pascal.

Однако совместимость C++Builder и Delphi этим не исчерпывается. В проектах C++Builder можно использовать не только библиотеку компонентов Delphi, но и код, написанный на Object Pascal, а также формы и модули Delphi. Поддерживается визуальное наследование форм и модулей данных, в том числе и созданных в Delphi. Эти возможности появились благодаря включению в C++Builder обоих компиляторов C++ и Object Pascal.

Это означает, что можно создавать общие проекты, используя оба средства разработки - и C++BuIder, и Delphi. Части одного приложения могут быть созданы с помощью двух средств, и теперь к работе над проектом можно привлекать разработчиков, использующих как Delphi, так и C++. Вовторых, и это очень важно, C++Builder может использовать компоненты, созданные для Delphi, а их за последние несколько лет создано огромное количество. Это богатство, накопленное разработчиками всего мира, сегодня способно удовлетворить самые причудливые запросы.

C++Builder предоставляет программисту широкие возможности повторного использования кода не только за счет наличия библиотеки компонентов, но и за счет поддержки стандарта ActiveX, что позволяет встраивать в приложения ActiveX-компоненты как сторонних производителей, так и созданные собственноручно с помощью самого C++Builder.

Немаловажным фактором, влияющим на популярность этих продуктов, является их открытость, заключающаяся в возможности создания с их помощью не только дополнительных компонентов и элементов ActiveX, улучшающих функциональность приложении, но и различных экспертов, редакторов свойств компонентов, улучшающих функциональность самой среды разработки.

Следует отметить, что эффективность разработки и отладки приложений достигается не только за счет использования удобных средств визуального проектирования форм (сейчас это не редкость), но и за счет, во-первых, высокой производительности самих компиляторов Borland и, во-вторых, так называемой инкрементной компиляции и компоновки исполняемого модуля (когда перекомпиляции и перекомпоновке подвергаются только те модули, в которые были внесены изменения).

В качестве среды для разработки выбран язык C++, входящий в состав программного комплекса Embarcadero.

Прежде всего, необходимо подчеркнуть, что оценивать достоинства и, в особенности, недостатки C++ необходимо в контексте тех принципов, на которых строился язык, и требований, которые к нему изначально предъявлялись.

C++ — чрезвычайно мощный язык, содержащий средства создания эффективных программ практически любого назначения, от низкоуровневых утилит и драйверов до сложных программных комплексов самого различного назначения. В частности:

Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщённое программирование, метапрограммирование (шаблоны, макросы).

Предсказуемое выполнение программ является важным достоинством для построения систем реального времени. Весь код, неявно генерируемый компилятором для реализации языковых возможностей (например, при преобразовании переменной к другому типу), определён в стандарте. Также строго определены места программы, в которых этот код выполняется. Это даёт возможность замерять или рассчитывать время реакции программы на внешнее событие.

Автоматический вызов деструкторов объектов при их уничтожении, причём в порядке, обратном вызову конструкторов. Это упрощает (достаточно объявить переменную) и делает более надёжным освобождение ресурсов (память, файлы, семафоры и т. п.), а также позволяет гарантированно выполнять переходы состояний программы, не обязательно связанные с освобождением ресурсов (например, запись в журнал).

Пользовательские функции-операторы позволяют кратко и ёмко записывать выражения над пользовательскими типами в естественной алгебраической форме.

Язык поддерживает понятия физической (const) и логической (mutable) константности. Это делает программу надёжнее, так как позволяет компилятору, например, диагностировать ошибочные попытки изменения значения переменной. Объявление константности даёт программисту, читающему текст программы дополнительное представление о правильном использовании классов и функций, а также может являться

подсказкой для оптимизации. Перегрузка функций-членов по признаку константности позволяет определять изнутри объекта цели вызова метода (константный для чтения, неконстантный для изменения). Объявление mutable позволяет сохранять логическую константность при использовании кэшей и ленивых вычислений.

Используя шаблоны, возможно создавать обобщённые контейнеры и алгоритмы для разных типов данных, а также специализировать и вычислять на этапе компиляции.

Возможность имитации расширения языка для поддержки парадигм, которые не поддерживаются компиляторами напрямую. Например, библиотека Boost.Bind позволяет связывать аргументы функций.

Возможность создания встроенных предметно-ориентированных языков программирования. Такой подход использует, например библиотека Boost.Spirit, позволяющая задавать EBNF-грамматику парсеров прямо в коде C++.

Используя шаблоны и множественное наследование можно имитировать классы-примеси и комбинаторную параметризацию библиотек. Такой подход применён в библиотеке Loki, класс SmartPrt которой позволяет, управляя всего несколькими параметрами времени компиляции, сгенерировать около 300 видов «умных указателей» для управления ресурсами.

Кроссплатформенность: стандарт языка накладывает минимальные требования на ЭВМ для запуска скомпилированных программ. Для определения реальных свойств системы выполнения в стандартной библиотеке присутствуют соответствующие возможности (например, std::numeric_limits <T>). Доступны компиляторы для большого количества платформ, на языке C++ разрабатывают программы для самых различных платформ и систем.

Эффективность. Язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы. Ни одна из языковых возможностей, приводящая к дополнительным накладным расходам, не является обязательной для использования — при необходимости язык позволяет обеспечить максимальную эффективность программы.

Имеется возможность работы на низком уровне с памятью, адресами.

Высокая совместимость с языком Си, позволяющая использовать весь существующий Си-код (код на Си может быть с минимальными переделками скомпилирован компилятором C++; библиотеки, написанные на Си, обычно могут быть вызваны из C++ непосредственно без каких-либо дополнительных затрат, в том числе и на уровне функций обратного вызова, позволяя библиотекам, написанным на Си, вызывать код, написанный на C++).

Теперь поговорим о недостатках C++.

Отчасти недостатки C++ унаследованы от языка-предка — Си, — и вызваны изначально заданным требованием возможно большей совместимости с Си. Это такие недостатки, как:

Синтаксис, провоцирующий ошибки:

Операция присваивания обозначается как =, а операция сравнения как ==. Их легко спутать, при этом операция присваивания возвращает значение, поэтому присваивание на месте выражения является синтаксически корректным, а в конструкциях цикла и ветвления появление числа на месте логического значения также допустимо, так что ошибочная конструкция оказывается синтаксически правильной. Типичный пример подобной ошибки:

if (x=0) { операторы }

Здесь в условном операторе по ошибке написано присваивание вместо сравнения. В результате, вместо того, чтобы сравнить текущее значение x с нулём, программа присвоит x нулевое значение, а потом интерпретирует его как значение условия в операторе if. Так как нуль соответствует логическому значению «ложь», блок операторов в условной конструкции не выполнится никогда. Ошибки такого рода трудно выявлять, но во многих современных компиляторах предлагается диагностика некоторых подобных конструкций.

Операции присваивания (=), инкрементации (++), декрементации (--) и другие возвращают значение. В сочетании с обилием операций это позволяет, хотя и не обязывает, создавать трудночитаемые выражения. Наличие этих операций в Си было вызвано желанием получить инструмент ручной оптимизации кода, но в настоящее время оптимизирующие компиляторы обычно генерируют оптимальный код и на традиционных выражениях. С другой стороны, один из основных принципов языков Си и C++ — позволять программисту писать в любом стиле, а не навязывать «хороший» стиль.

Макросы (#define) являются мощным, но опасным средством. Они сохранены в C++ несмотря на то, что необходимость в них, благодаря шаблонам и встроенным функциям, не так уж велика. В унаследованных стандартных Си-библиотеках много потенциально опасных макросов.

Некоторые преобразования типов неинтуитивны. В частности, операция над беззнаковым и знаковым числами выдаёт беззнаковый результат.

C++ позволяет пропускать break в ветви оператора switch с целью последовательного выполнения нескольких ветвей. Такой же подход принят в языке Java. Есть мнение, что это затрудняет понимание кода. Например, в языке C# необходимо всегда писать либо break, либо использовать goto case N для явного указания порядка выполнения.

Препроцессор, унаследованный от Си, очень примитивен. Это приводит с одной стороны к тому, что с его помощью нельзя (или тяжело) осуществлять некоторые задачи метапрограммирования, а с другой, вследствие своей примитивности, он часто приводит к ошибкам и требует много действий по обходу потенциальных проблем. Некоторые языки программирования (например, Scheme и Nemerle) имеют намного более мощные и более безопасные системы метапрограммирования (также называемые макросами, но мало напоминающие макросы Си/C++).

Плохая поддержка модульности (по сути, в классическом Си модульность на уровне языка отсутствует, её обеспечение переложено на компоновщик). Подключение интерфейса внешнего модуля через препроцессорную вставку заголовочного файла (#include) серьёзно замедляет компиляцию при подключении большого количества модулей (потому что результирующий файл, который обрабатывается компилятором, оказывается очень велик). Эта схема без изменений скопирована в C++. Для устранения этого недостатка, многие компиляторы реализуют механизм прекомпиляции заголовочных файлов (англ. Precompiled header).

К собственным недостаткам C++ можно отнести:

Сложность и избыточность, из-за которых C++ трудно изучать, а построение компилятора сопряжено с большим количеством проблем. В частности:

Многие конструкции C++ позволяют делать то же самое, что и конструкции Си, также присутствующие в C++. Это иногда сбивает с толку новичков. Например, приведение типов при помощи dynamic_cast позволяет привести указатель или ссылку строго в пределах иерархии классов. Это делает код более надёжным, декларативным и позволяет находить приведения в пределах иерархии при помощи инструментов типа grep. Однако вследствие требования высокой степени совместимости с Си старое приведение типов всё ещё поддерживается.

Иногда шаблоны приводят к порождению кода очень большого объёма. Для снижения размера машинного кода можно специальным образом подготавливать исходный код. Другим решением является стандартизованная ещё в 1998 году возможность экспорта шаблонов. Некоторые авторы считают, что её трудно реализовать и поэтому она доступна не во всех компиляторах. «Раздувание» машинного кода вследствие использования шаблонов часто преувеличивается, и современные компиляторы во многих случаях успешно устраняют это явление.

Метапрограммирование на основе шаблонов C++ сложно и при этом ограничено в возможностях. Оно состоит в реализации средствами шаблонов C++ интерпретатора примитивного функционального языка программирования, выполняющегося во время

компиляции. Сама по себе данная возможность весьма привлекательна, но такой код весьма трудно воспринимать и отлаживать. Менее распространённые языки Lisp/Scheme, Nemerle имеют более мощные и одновременно более простые для восприятия подсистемы метапрограммирования. Кроме того, в языке D реализована сравнимая по мощности, но значительно более простая в применении подсистема шаблонного метапрограммирования.

Явная поддержка функционального программирования присутствует только в будущем стандарте c++0x. Данный пробел устраняется различными библиотеками (Loki, Boost), использующими средства метапрограммирования для расширения языка функциональными конструкциями (например, поддержкой лямбд/анонимных методов), но качество подобных решений значительно уступает качеству встроенных в функциональные языки решений. Такие возможности функциональных языков, как сопоставление с образцом, вообще крайне сложно эмулировать средствами метапрограммирования.

Некоторые считают недостатком языка C++ отсутствие встроенной системы сборки мусора. С другой стороны, средства C++ позволяют реализовать сборку мусора на уровне библиотеки. Противники сборки мусора полагают, что RAII является более достойной альтернативой. C++ позволяет пользователю самому выбирать стратегию управления ресурсами.

### 2.3. Разработка формы входных и выходных данных

Одной из главных задач любого программного обеспечения, будь то программа, база данных, сайт или иное обеспечение, является ввод-вывод данных. Форма входных данных может быть различной, но, как правило, в большинстве случаев это текст, который в дальнейшем подлежит обработке, или же числовые данные определенного формата.

В данной работе почти все входящие данные имеют тип String. Так как никаких вычислений в программе не предполагается, соответственно, это упрощает процесс планирования кода. Что касается запретов на ввод определенных символов, они легко реализуются и с данными строкового типа. Список данных типа Integer (целочисленные) будет приведен ниже.

Современное программное обеспечение, предназначенное для пользователя (в отличие от системного ПО), должно быть ориентировано не только на правильность и точность поставленных задач, но и на удобство использования. С этой целью программа должна иметь внятный, легкий в использовании интерфейс. Ввод данных практически всегда осуществляется с помощью такого компонента, как однострочный редактор (edit). Он удобен в использовании, с его помощью легко упорядочивать данные программным способом. Из родственных компонентов можно отметить многострочные поля ввода Memo,

однако у них есть существенный недостаток – данные, которые вводятся с их помощью, не подлежат сортировке, фильтрации и индексированию, т.к. требуют намного больших вычислительных мощностей. Это нецелесообразно. Иногда на помощь при вводе данных также используют компоненты ListBox и ComboBox, данные в которых, как правило, предустановлены и пользователь только выбирает подходящие для себя значения. В настоящей разработке такие компоненты также применяются. Иногда роль компонентов, позволяющих осуществить ввод данных, могут взять на себя CheckBox и RadioButton, но это большая редкость – в этом случае речь идет о простом выборе из ограниченного набора каких-либо значений. Довольно часто применяется ввод данных в компонент Table, однако у него специфические свойства, ограничивающие его применение.

Вывод данных можно осуществлять в компоненты Edit, Memo, Label, Table и т.п.

Что касается типов данных Integer, они присутствуют в данной разработке в Unit2, Unit4, Unit5, Unit13, Unit16 и объявлены в виде глобальных переменных.

### 2.4. Разработка алгоритма решения задачи

Рисунок 2.1. Общая схема алгоритма работы программы

Примеры алгоритмов решения различных задач разработки:

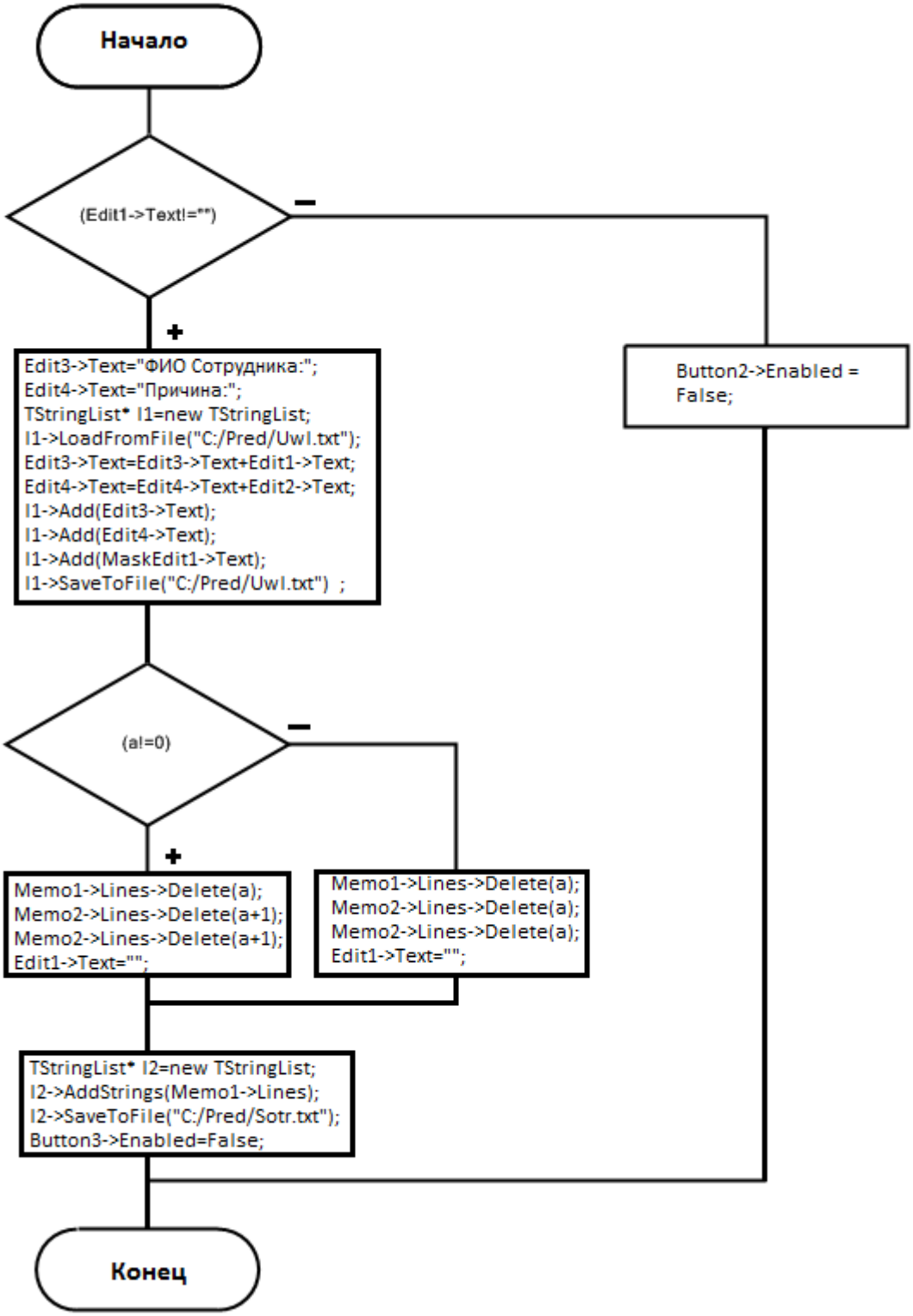



Рисунок 2.2. Алгоритм работы формы «Увольнение сотрудника»

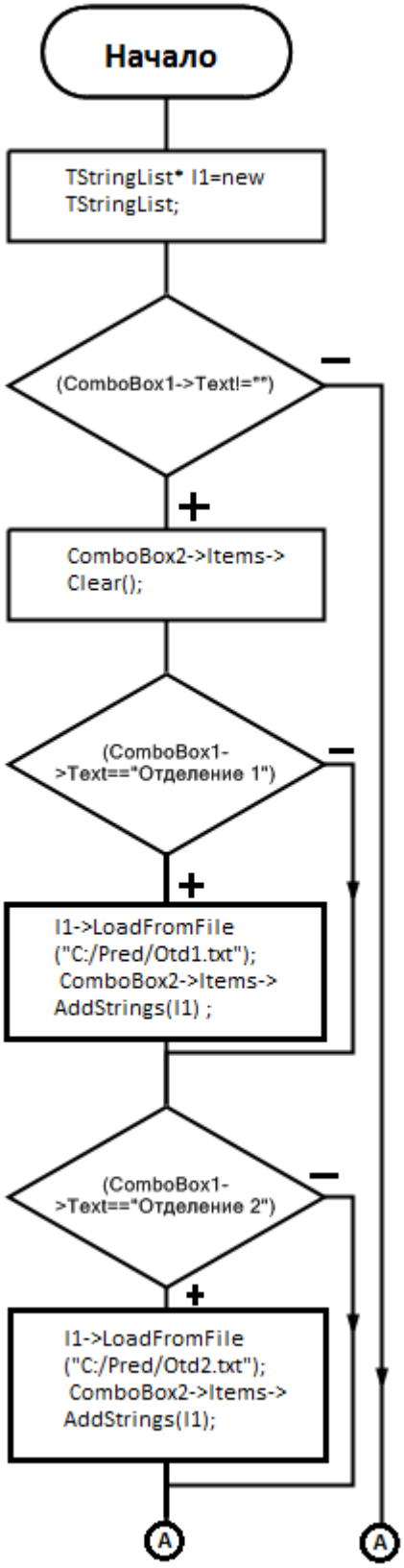



Рисунок 2.3.Добавление должности в подразделение

## 2.5. Разработка интерфейсной части

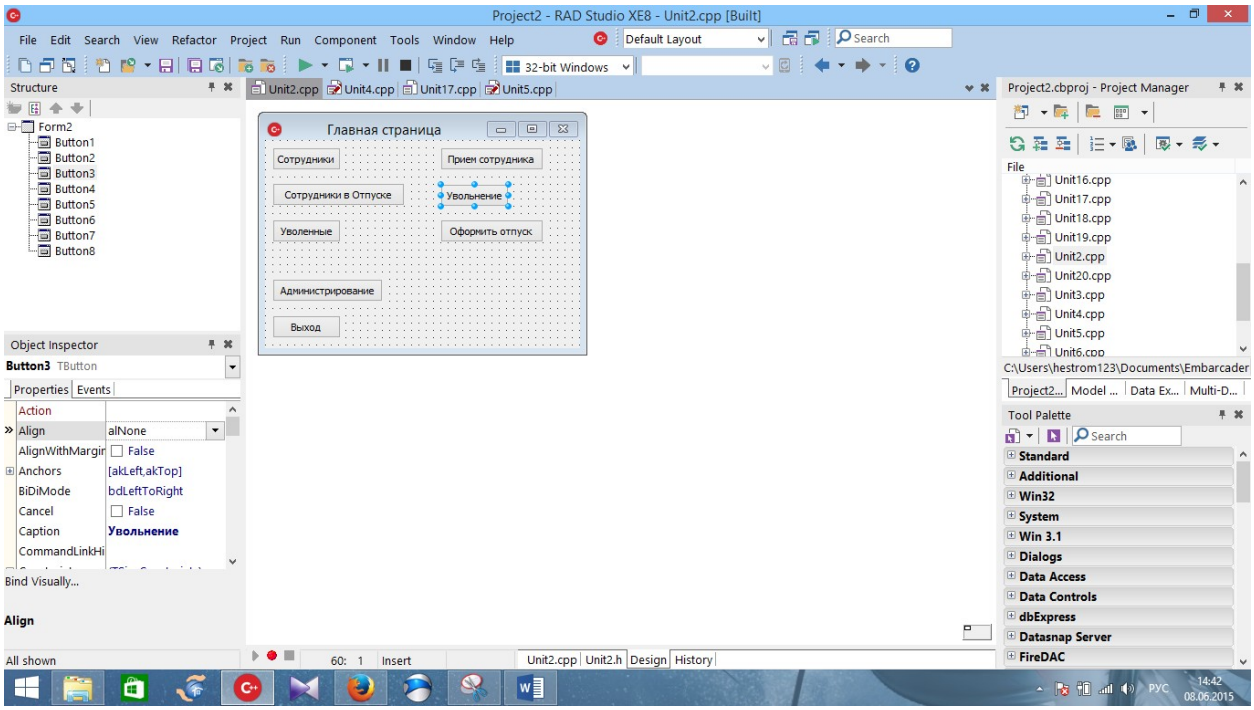



Рисунок 2.4. Проектирование главного окна приложения.

Любая программа требует обязательного создания форм ввода/вывода данных – эта работа выполняется сразу же после разработки алгоритма. В объектно-ориентированном программировании, разумеется, возможны иные подходы, но проектирование кода после создания интерфейса выглядит наиболее логичным и целесообразным.

На рисунке выше представлен проект главной формы приложения, созданный в среде разработки Embarcadero. В правой части окна расположена панель инструментов с визуальными и невизуальными компонентами.

Установка компонентов на форму выполняется просто – достаточно выбрать необходимый и поместить его на рабочее поле формы, обозначенное серым фоном. После установки любого компонента автоматически в левой нижней части окна (по умолчанию) появляется Инспектор Объектов, который показывает свойства выделенного объекта. Если ни один объект не выделен, в Инспекторе отображаются свойства текущей формы.

Типичными свойствами формы являются BorderIcons (кнопки вверху формы, от их набора зависит иногда сам тип формы), Caption (видимый заголовок формы), BorderStyle (стиль границ – от изменяемых до диалогового окна, когда размер формы остается постоянным и ее нельзя свернуть и развернуть на весь экран), а также Icon (логотип приложения в левом верхнем углу формы).

Основными компонентами главной формы приложения являются кнопки (компонент Button). Основным событием, которое связано с кнопкой, является щелчок мышью (событие OnClick). Описание программного кода главной формы дано в приложении Б (Руководстве программиста), внешний вид модулей в среде Embarcadero – в следующей теме.

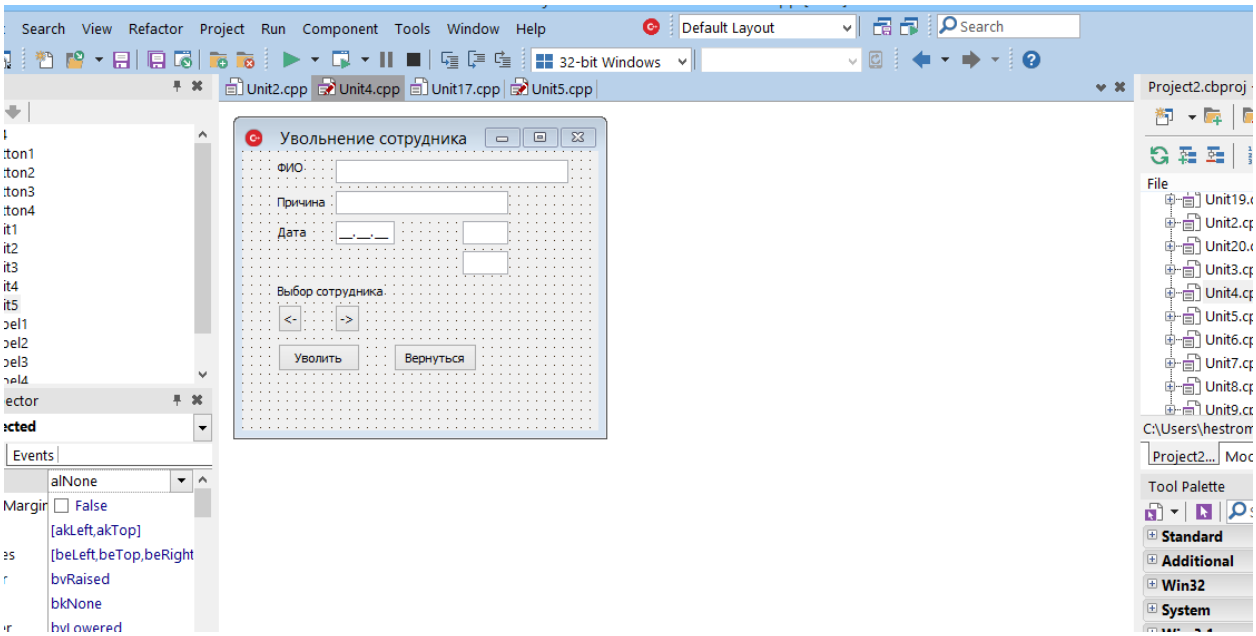



Рисунок 2. 5.Создание формы «Увольнение сотрудника».

В данной форме появляются такие компоненты, как Edit и Label. Свойства компонента Label (Надпись) особого смысла перечислять нет – он не задействован в программе и играет роль простой надписи. Об edit следует рассказать чуть подробнее – во всех формах, этой и тех, о которых речь идет далее, они используются для ввода данных.

У этого объекта достаточное количество интересных и полезных для программиста свойств. То, что видит пользователь – иногда результат многочасовых усилий программистов. Например, свойство MaxLength – ограничитель количества вводимых символов. Смысл этого ограничения будет описан позже.

Свойство Font регулирует параметры шрифта – размер, цвет, начертание.

Маска ввода – данные вводятся согласно определенным правилам.

Но в основном свойства компонента Edit регулируются программным способом.

Также в данной работе встречается компонент ListBox, едва ли не главным свойством которого является Items. Собственно говоря, это не что иное, как набор данных, которые отображаются в компоненте. Источником данных может быть что угодно – например, какой-нибудь список из текстового файла. Это достаточно легко программируется. У объекта имеются свойства, делающие его возможности разнообразными – к примеру, Multiselect, когда можно выделить сразу несколько строк данных (по умолчанию этот параметр отключен). Sorted – свойство, означающее сортировку данных в алфавитном порядке, однако это действие требуется далеко не всегда.

Формы создаются просто – в меню File нужно выбрать New-Form. Ссылки на новую форму появятся автоматически, стоит хотя бы раз упомянуть ее в коде. Список всех форм

моно посмотреть в Менеджере проекта (правое верхнее вспомогательное окно рабочей среды Embarcadero).

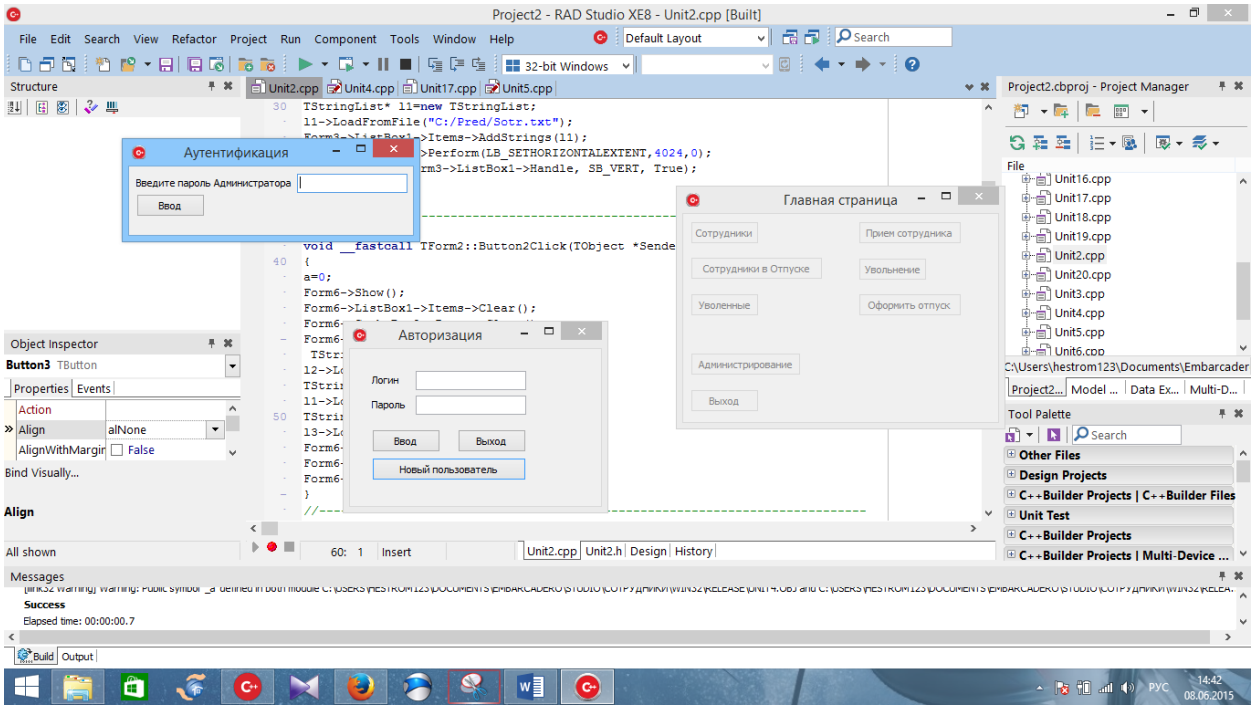



Рисунок 2.6. Создание форм регистрации и аутентификации.

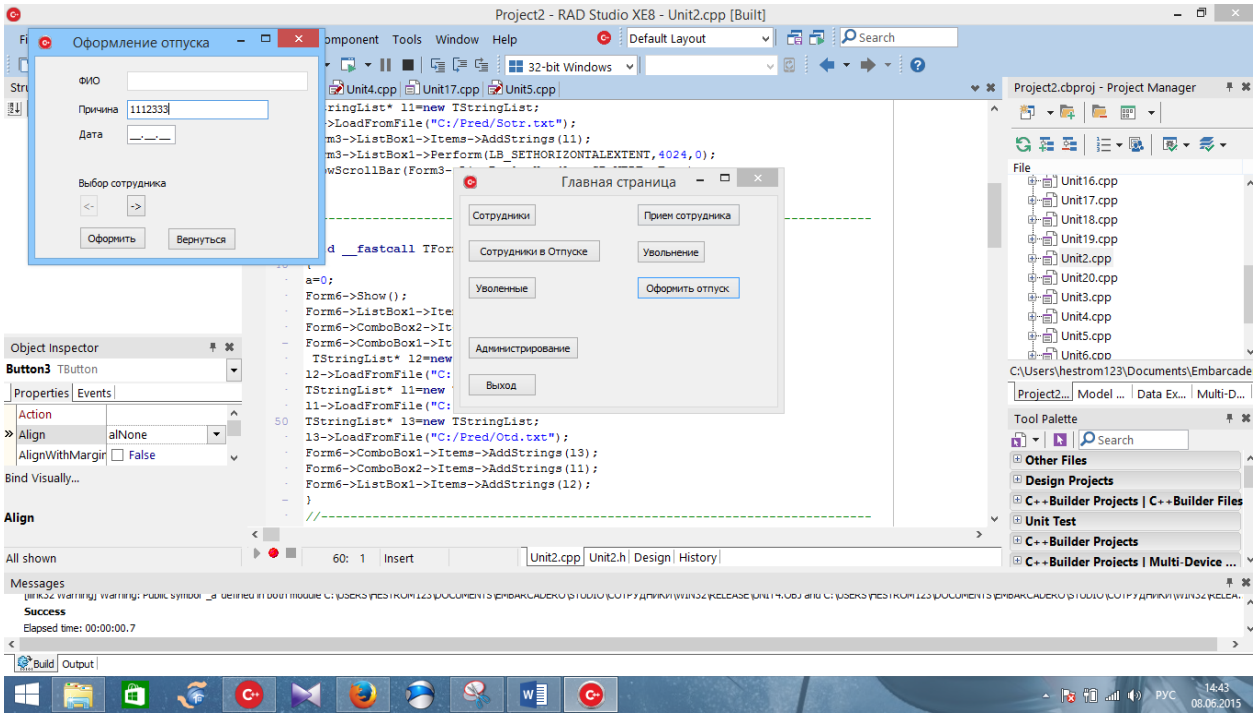



Рисунок 2.7. Создание формы «Оформление отпуска»

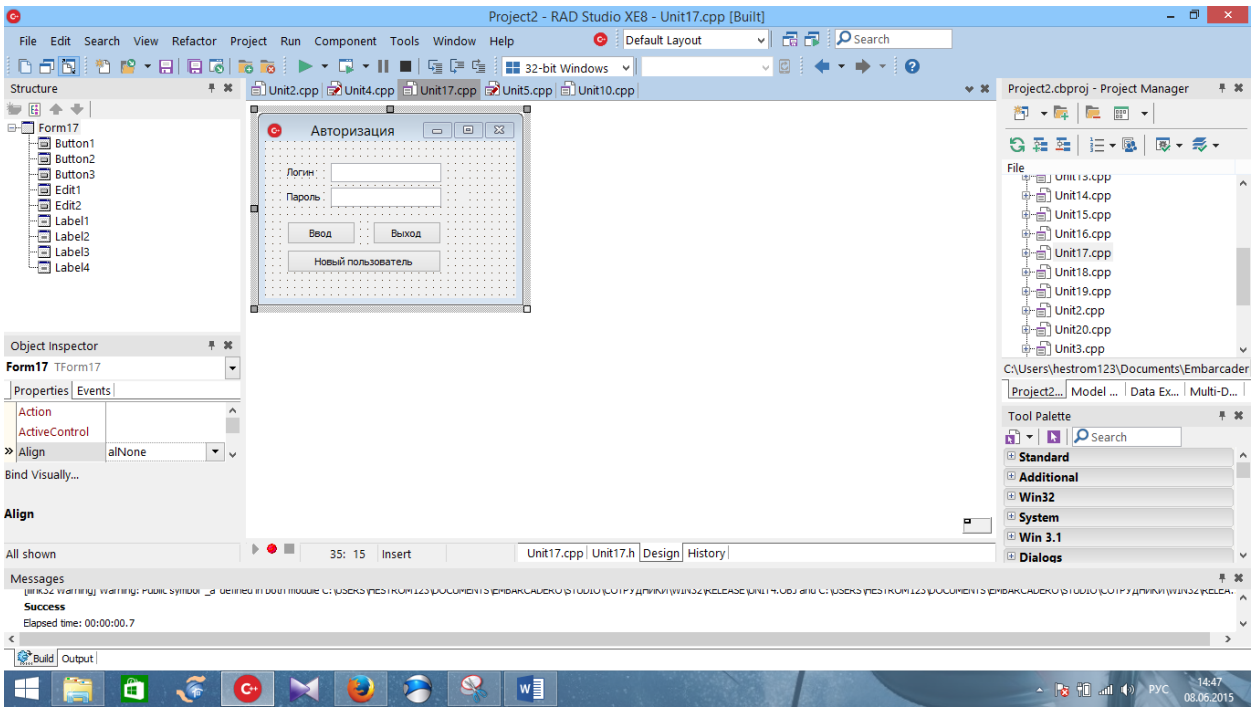



Рисунок 2.8. Создание формы «Авторизация».

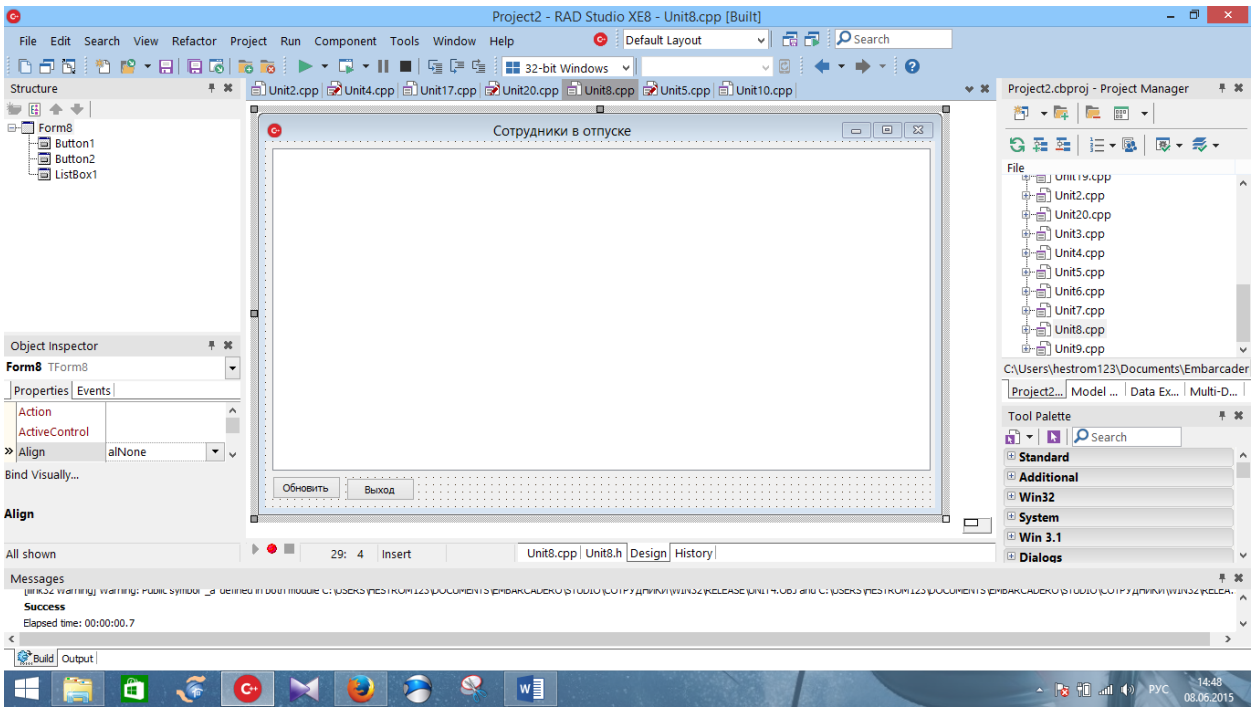



Рисунок 2.9. Создание формы «Сотрудники в отпуске»

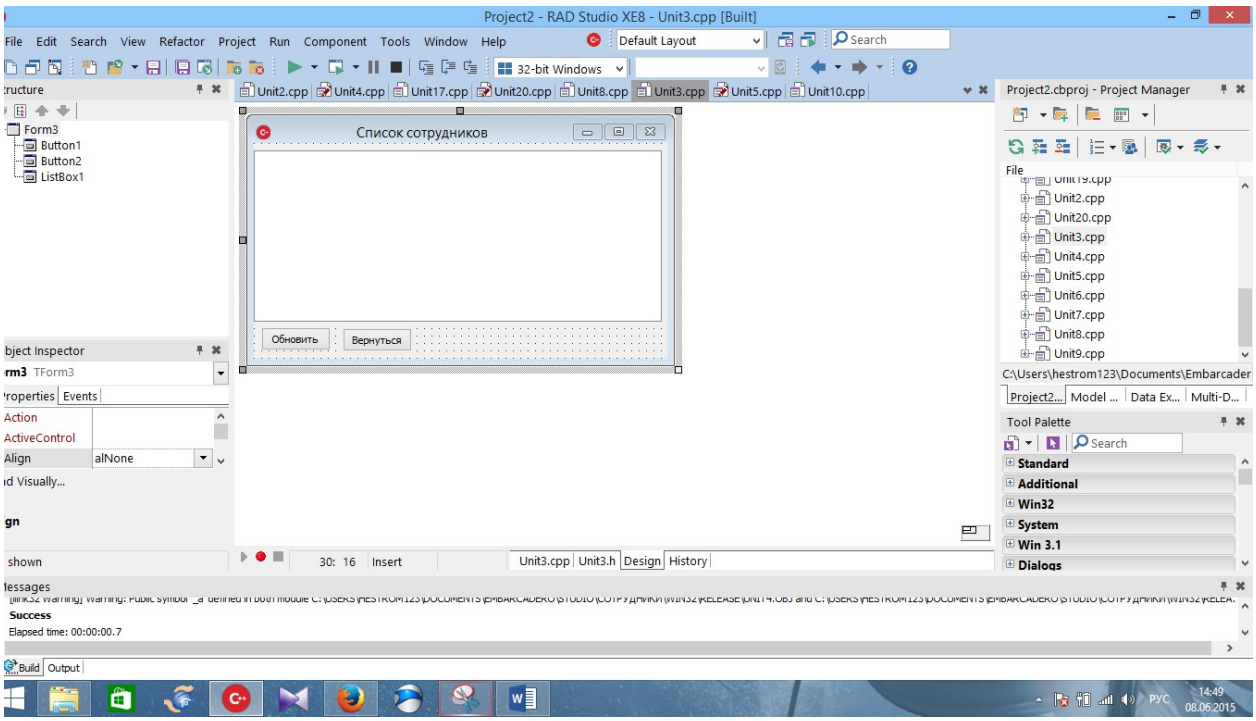



Рисунок 2.10. Создание формы «Список сотрудников»

## 2.6. Описание модулей программы

```
{
if (Edit1->Text!="")
{
Edit3->Text="ФИО Сотрудника:";
Edit4->Text="Причина:";
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Uwl.txt");
Edit3->Text=Edit3->Text+Edit1->Text;
Edit4->Text=Edit4->Text+Edit2->Text;
l1->Add(Edit3->Text);
l1->Add(Edit4->Text);
l1->Add(MaskEdit1->Text);
l1->SaveToFile("C:/Pred/Uwl.txt")  ;
ListBox1->Items->Delete(a);
Edit1->Text="";
TStringList* l2=new TStringList;
l2->AddStrings(ListBox1->Items);
l2->SaveToFile("C:/Pred/Sotr.txt");
Button3->Enabled=False;
ListBox1->Items->Clear();
Form4->Close();
}
else
{
MessageBox(NULL,L"Вы совершили ошибку",L"Ошибка",MB_OK );}
```

Рисунок 2.11. Фрагмент листинга программного кода

Как уже было сказано выше, данные в программе записываются в специально организованные файлы текстового формата, которые находятся в папке C:\Pred. Список файлов: Admin.txt, Otd1.txt, Otd2.txt, Otd3.txt, Otd4.txt, Otp.txt, Sotr.txt, Uwl.txt.

Соответственно из этих же файлов данные при необходимости и извлекаются. Для повышения безопасности данных файлы можно подвергнуть шифрованию, а в обработку событий добавить криптоалгоритм, однако такая задача поставлена не была, тем не менее, такая возможность существует.

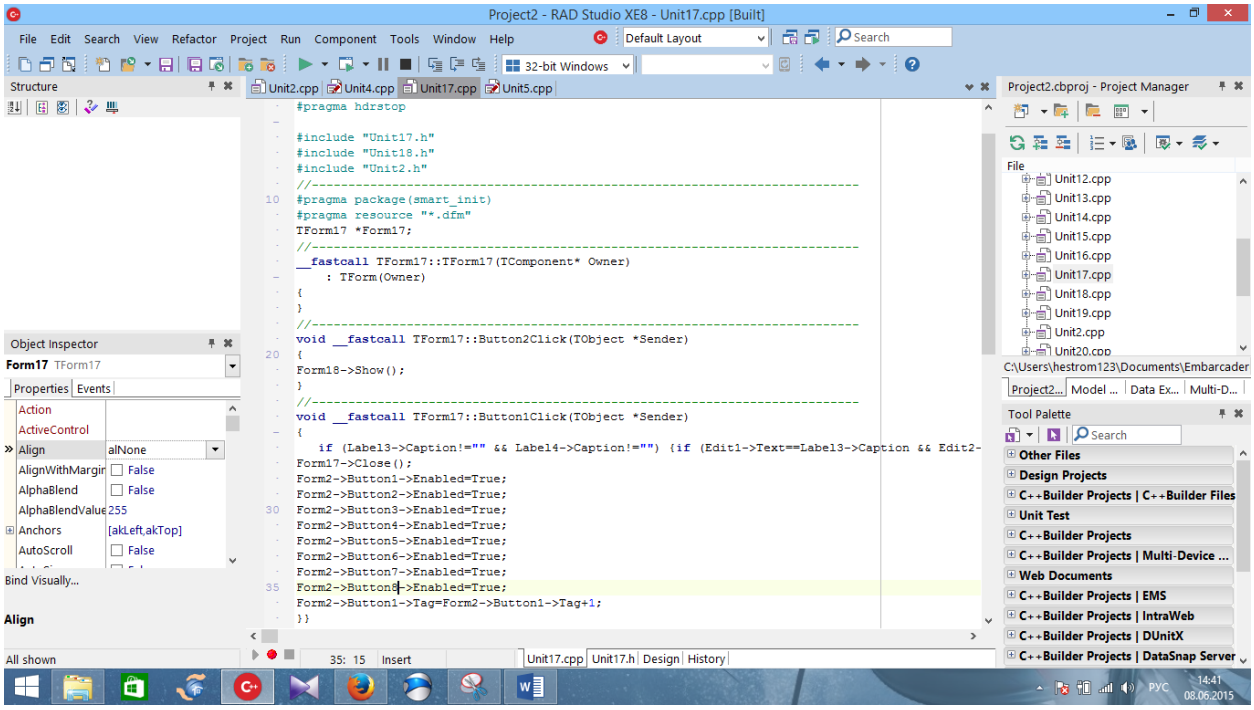



Рисунок 2.12. Описательная часть формы в виде программного кода

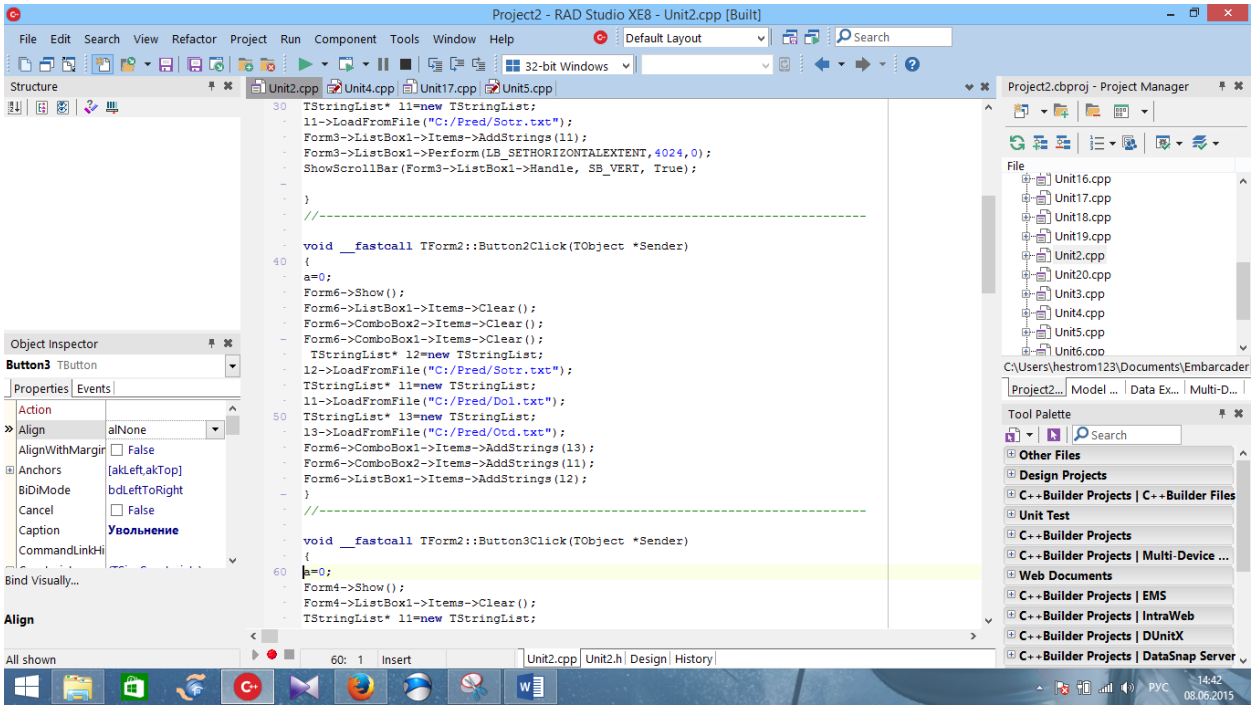



Рисунок 2.13. Фрагмент кода, описывающий загрузку данных из файла

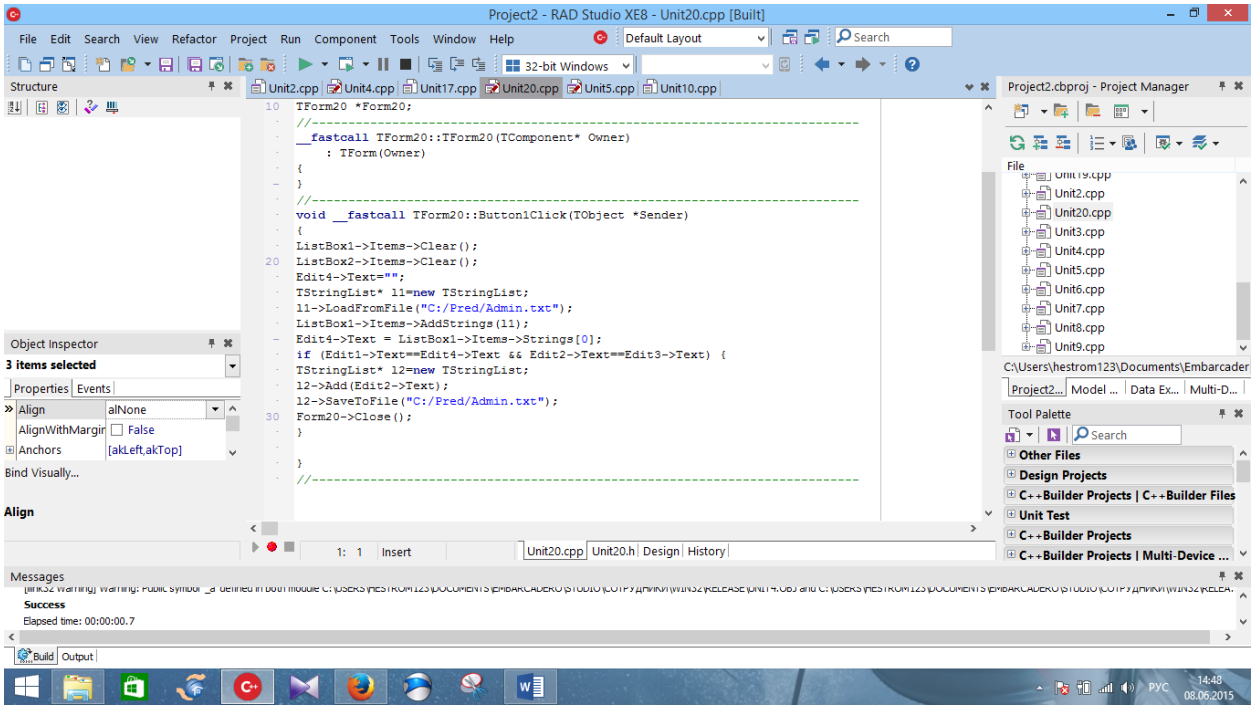



Рисунок 2.14. Фрагмент кода, регламентирующий работу с паролем администратора

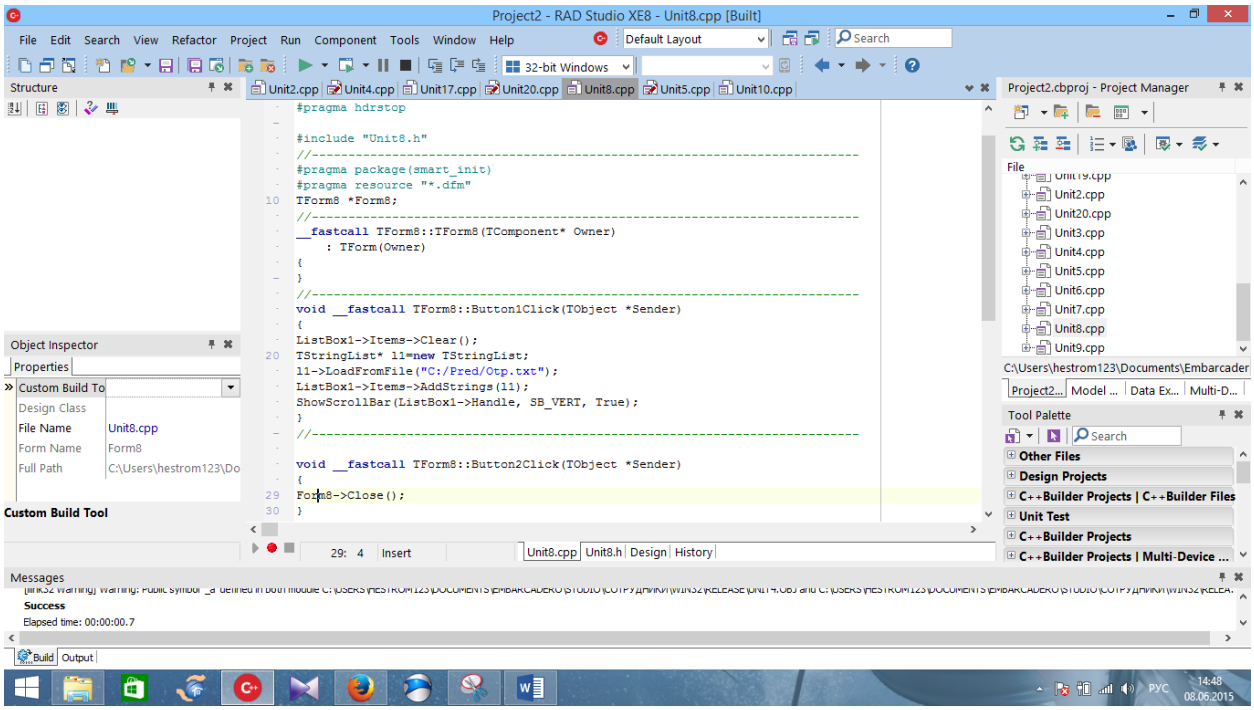



Рисунок 2.15. Загрузка списка сотрудников в компонент ListBox

Скриншоты из среды разработки приведены в качестве примера. Действия, производимые в различных модулях программы, т.н. Units, достаточно типичны. Листинг отдельных модулей приведен в приложении Б (в руководстве программиста) и снабжен комментариями, позволяющими понять логику работы программы.

## 2.7. Описание процесса отладки программы

### 2.7.1. Тестирование

ПРОТОКОЛ
тестовых испытаний программы «Сотрудники»

Настоящий протокол составлен по результатам тестирования на базе программно-технических средств Силина А.С., проведенного в период с 28 мая 2015 г. по 29 мая 2015 г.

Условия, в которых проводилось тестирование:

Техническая среда тестирования — ПК с операционной системой Windows 8.1 и средой разработки Embarcadero RAD Studio XE8.

Краткие характеристики ПК:

- Процессор AMD A8-5557M 2.10HZ;
- Видеокарта AMD Radeon HD 8750M;
- ОЗУ память 6000 MB Kingston DDR2;
- Жесткий диск Western Digital WD5000AAKX.

Выводы по результатам тестирования:

При создании интерфейса программы никаких проблем обнаружен не было. Тестирование первого алгоритма ошибок не выявило.

Если попытаться ввести в поле запрещенный символ(число или символ), то появится сообщение об ошибке:

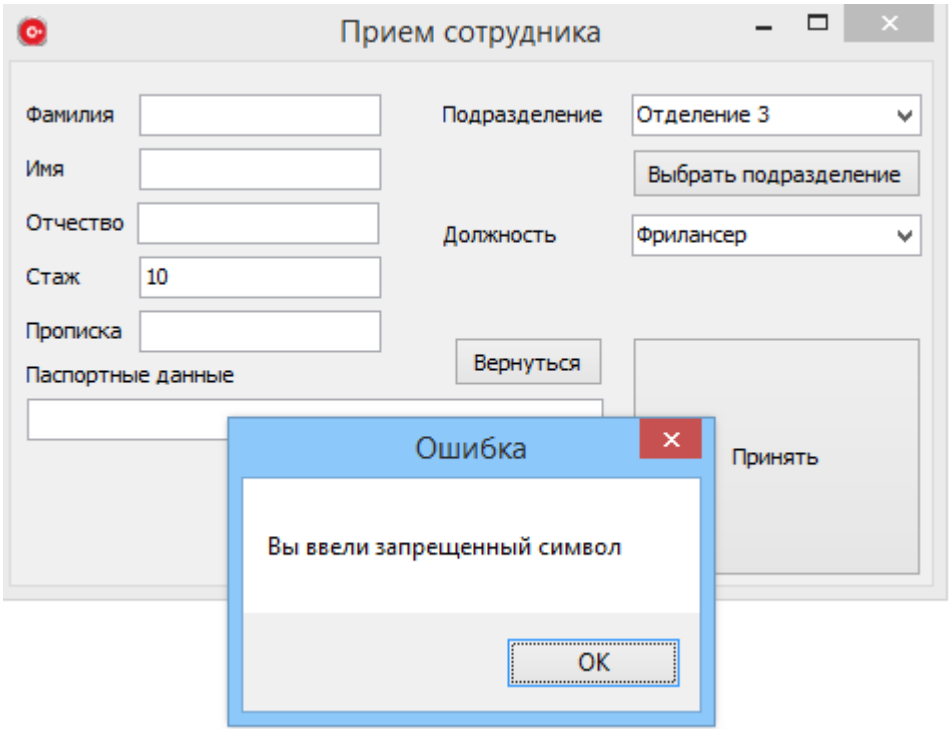



Рисунок 2.16. Сообщение об ошибке в результате тестирования (введен запрещенный символ)

На полях стоят различные ограничения, не позволяющие ввести количества символов больше разрешенного. Также нельзя оставлять поля пустыми, при нажатии на кнопку «Принять» появится сообщение о том, что одно из полей пустое.

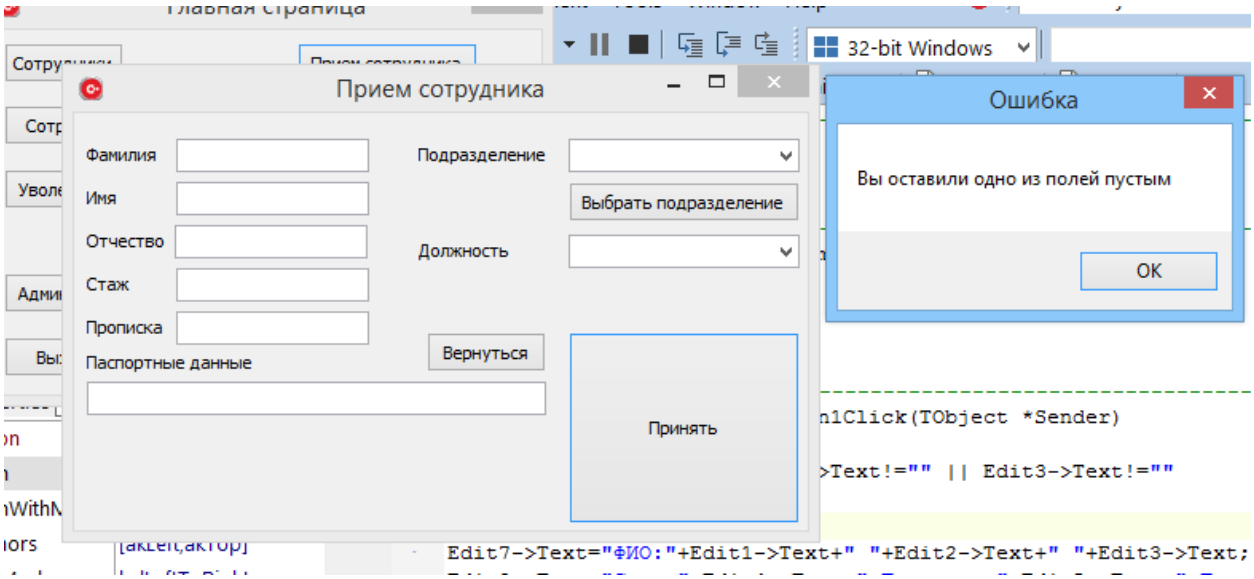



Рисунок 2.17. Ошибка при заполнении формы – одно поле не содержит данных

В данном алгоритме неисправностей не выявлено.

Следующее испытание было проведено над формой «Увольнение». На форме нельзя менять текст в объекте Edit1 напрямую, так как его доступность ограничена. Поле причина ограничино 50 символами. Поле Дата ограничено 6 символами и в поле можно вводить только цифры.

Если попытаться оставить поле ФИО пустым, то появится MessageBox с сообщением об ошибке. После чего форма снова станет доступной.

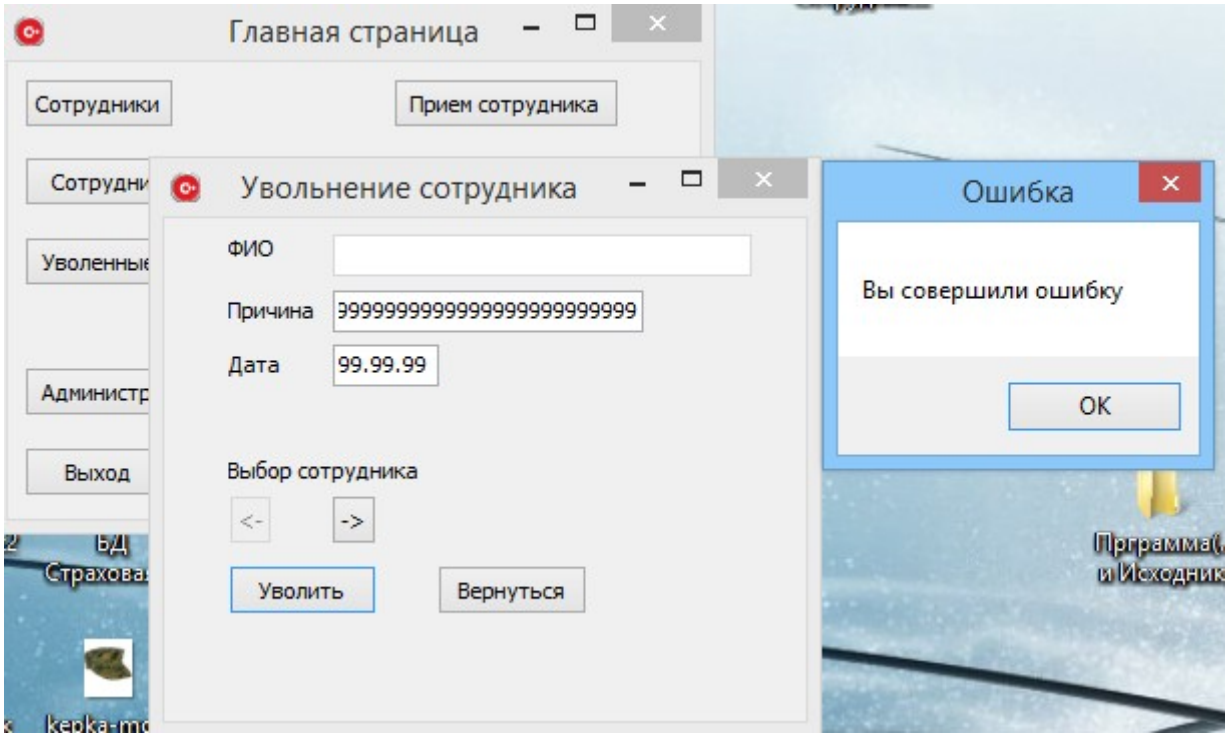



Рисунок 2.18. Попытка удалить сотрудника, не выбрав его

В данном алгоритме неисправностей не выявлено.

### 2.7.2. Контрольный пример

Суть примера заключается в том, что необходимо занести запись в базу данных программы и проверить, появилась ли эта запись в файле БД текстового формата и отображается ли она в специальной форме «Список сотрудников». 1 этап: авторизация.

Для входа в программу необходимо авторизоваться. Для этого должна быть или учетная запись администратора или учетная запись пользователя. Запись пользователя можно создать при наличии пароля администратора.

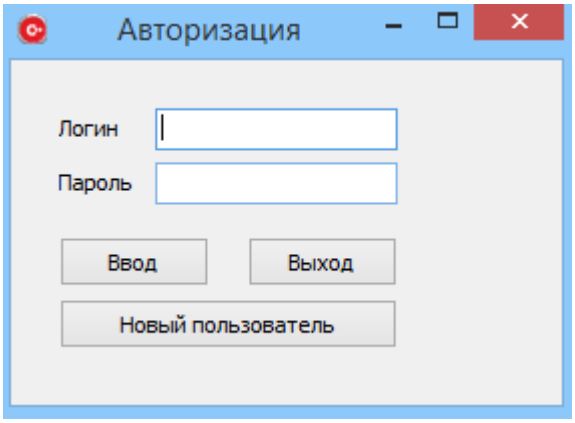


Рисунок 2.19. Окно авторизации пользователя

2 этап. Добавление сотрудника в базу

В главной форме нажатием кнопки «Принять сотрудника» открывается форма «Принять сотрудника».

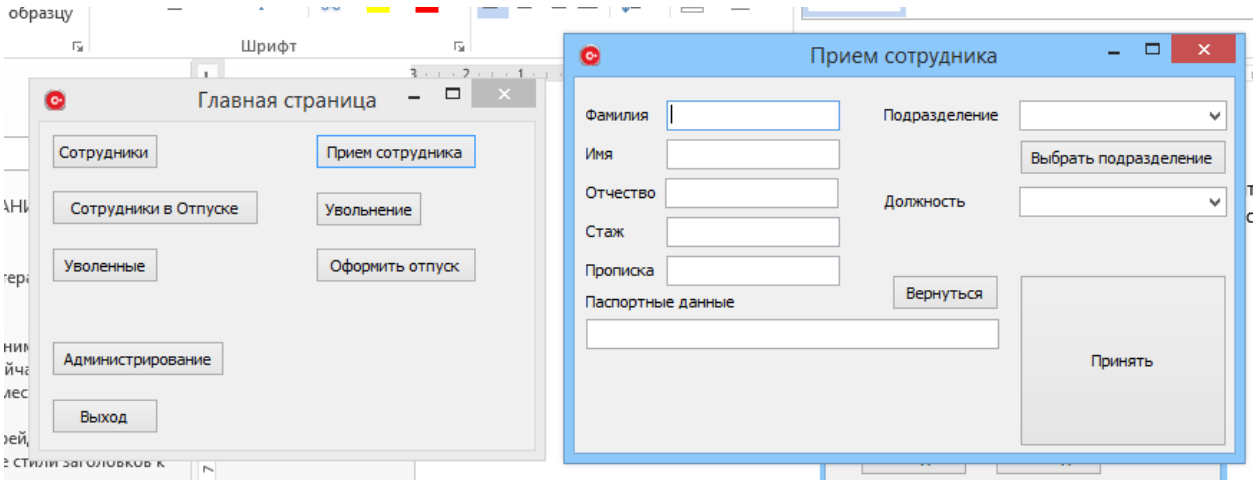



Рисунок 2.20. Процедура создания новой записи в базе данных программы

В пустые поля на форме Прием сотрудника заносим данные.

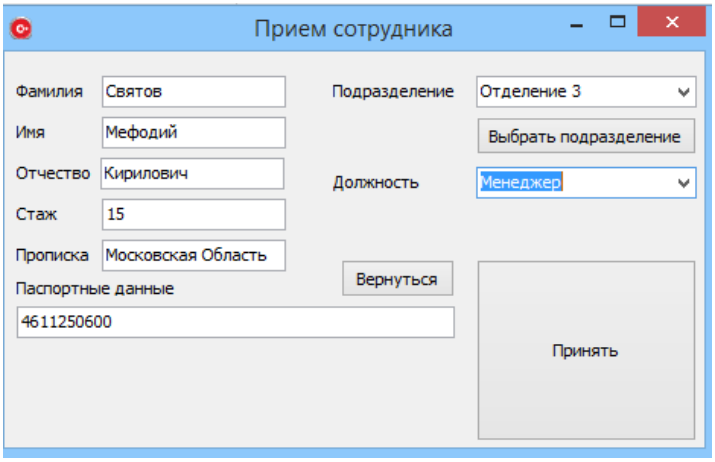



Рисунок 2.21. Образец правильного заполнения формы

Далее после нажатия на кнопку принять информация занесется в файл.

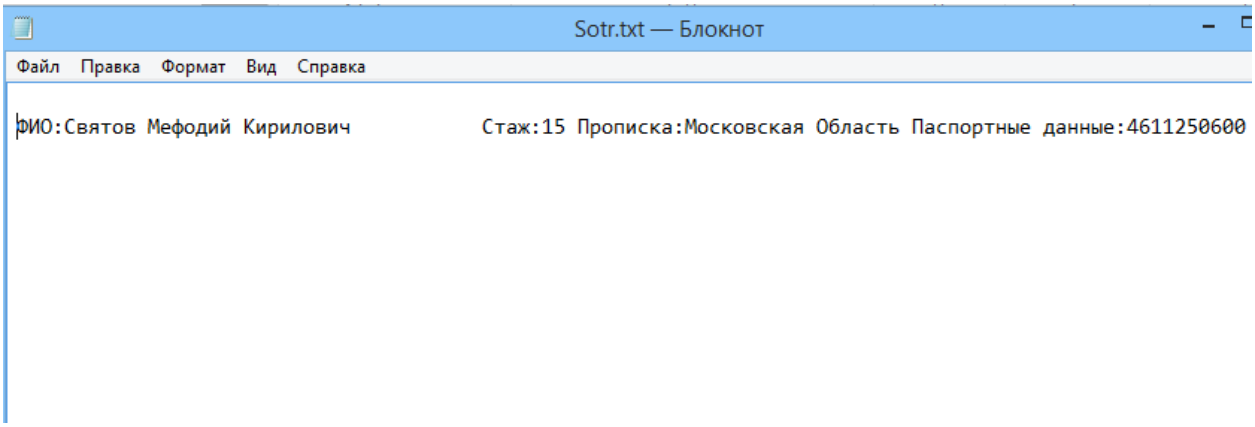



Рисунок 2.22. Новая запись в базе данных программы

3.Этап. Проверка формы «Список сотрудников».

Просмотрим информацию о сотрудниках через форму Сотрудники. Для этого на главной форме нажимаем кнопку Сотрудники.

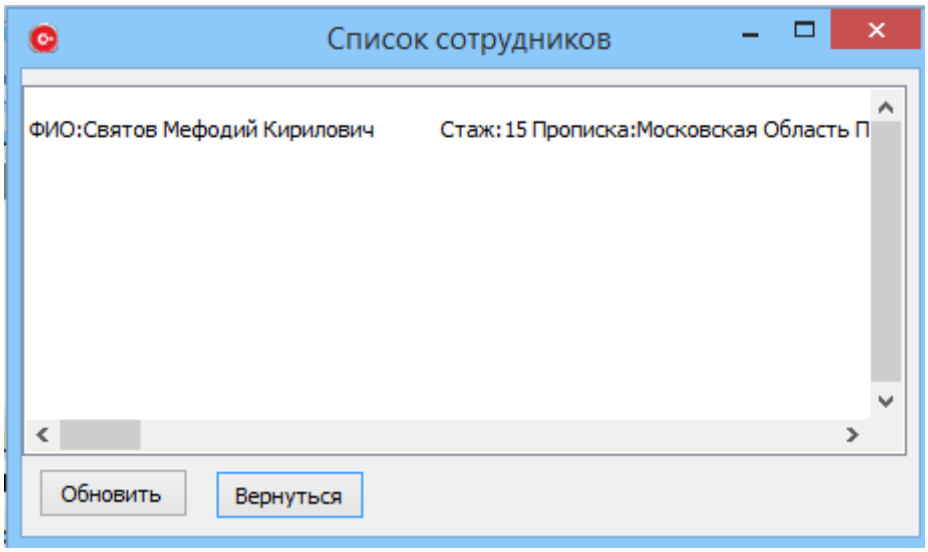



Рисунок 2.23. Прогнозируемый результат контрольного примера

При нажатии кнопки «Вернуться» форма «Список сотрудников» закроется.

### 2.8. Разработка мер защиты информации от несанкционированного доступа

Программа защищена с помощью аккаунта, который при использовании программы необходимо создать для каждого пользователя.

Если пользователь не регистрируется в системе, кнопки главной формы будут недоступны и никакие действия в программе произвести нельзя. Ниже приведен скриншот, иллюстрирующий этот момент.

Регистрация нового пользователя возможна только с участием лица, обладающего административными правами. Нажатие на кнопку регистрации вызывает окно ввода административного пароля. Обойти это действие невозможно. Пароль устанавливается непосредственно при разработке программы. Он стандартен и меняется с помощью опции «Изменить пароль администратора» в окне «Администрирование». Это действие желательно выполнить при первом же запуске программы.

В качестве меры безопасности рекомендуется установка программы с помощью инсталлятора. Наличие исходного кода не является необходимостью на компьютере сотрудника отдела кадров. Исходный код необходим только администратору, этот код идет в пакете поставки.

Т.к. данный программный продукт не использует системные функции Windows, использование инсталлятора не является полной необходимостью и лишь демонстрирует стремление разработчика к созданию качественного продукта.

Также программа широко использует защиту от неправильного ввода данных. В работе всегда нужно учитывать «человеческий фактор», который проявляется в том, что сотрудник, который занимается обработкой данных, по случайности, рассеянности или иной причине может ввести в поля данные, которые или не соответствуют принятым правилам (например, в фамилии указаны цифры), или способны вызвать аварийное завершение работы программы.

Такая защита в программе предусмотрена, в частности, обязательно соблюдение определенных правил при заполнении полей Edit с различными типами данных.

В поле «Стаж» невозможен ввод букв. Поле «Дата» (всех окон) представляет собой маску, полностью исключающую неправильный ввод данных.

Еще один компонент безопасности – ограничение количества вводимых символов. Как правило, ограничения эти следует устанавливать, исходя из разумных пределов. Фамилия из более чем 20 букв – исключительная редкость. На практике достаточно и 15 символов. Такие ограничения в программе есть.

Одним из основополагающих законов баз данных – никаких пустых записей в таблицах быть не должно. Пустые строки способны вызвать ошибки при обработке данных и доставляют одни неприятности. Данный программный продукт не позволяет вносить пустые строки в БД: как минимум, хотя бы одно поле при вводе данных должно быть заполнено.

# 3. ОХРАНА ТРУДА И БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

## 3.1. Охрана труда.

Программист в силу специфики своей профессиональной деятельности вынужден подавляющую часть своего рабочего времени проводить за компьютером, а это значит, что на него воздействует множество негативных факторов, способных пагубно сказаться на его здоровье и работоспособности. В целях минимизации вредного воздействия работающего оборудования и иных специфичных воздействий, обусловленных особенностями труда программиста, существуют санитарные нормы и требования к условиям, в которых программист должен работать.

На программиста могут воздействовать опасные и вредные производственные факторы:

Физические:

а) Повышенные уровни электромагнитного излучения.

б) Повышенные уровни рентгеновского излучения.

в) Повышенные уровни ультрафиолетового излучения.

г) Повышенный уровень инфракрасного излучения.

д) Повышенный уровень статического электричества.

е) Повышенные уровни запыленности воздуха рабочей зоны.

ж) Повышенное содержание положительных аэроионов в воздухе рабочей зоны.

з) Пониженное содержание отрицательных аэроионов в воздухе рабочей зоны.

и) Пониженная или повышенная влажность воздуха рабочей зоны.

к) Пониженная или повышенная подвижность воздуха рабочей зоны.

л) Повышенный уровень шума.

м) Повышенный или пониженный уровень освещенности.

н) Повышенный уровень прямой блесткости.

о) Повышенный уровень отраженной блесткости.

п) Повышенный уровень ослепленности.

р) неравномерность распределения яркости в поле зрения.

с) повышенная яркость светового изображения;

т) повышенный уровень пульсации светового потока;

у) повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;

Химические:

а) Повышенное содержание в воздухе рабочей зоны двуокиси углерода, озона, аммиака, фенола, формальдегида и полихлорированных бифенилов.

Психофизиологические:

а) Напряжение зрения.

б) Напряжение внимания.

в) Интеллектуальные нагрузки.

г) Эмоциональные нагрузки.

д) Длительные статические нагрузки.

е) Монотонность труда.

ж) Большой объем информации, обрабатываемой в единицу времени.

з) Нерациональная организация рабочего места.

Биологические:

а) Повышенное содержание в воздухе рабочей зоны микроорганизмов.

### 3.2. Техника безопасности

- К самостоятельной работе с компьютером, ноутбуком, принтером, ксероксом, сканером, плазменной панелью, LCD-экраном и другой оргтехникой допускаются лица, достигшие 18 летнего возраста, прошли медицинский осмотр, инструктаж по охране труда, не имеющие противопоказаний по состоянию здоровья.
- Осветительные установки должны обеспечивать равномерное освещение и не должны образовывать ослепляющих отблесков на клавиатуре, а также на экране монитора по направлению глаз.
- При работе с компьютером, принтером, ксероксом и другой периферийной техникой не допускается расположение рабочего места в помещениях без естественного освещения, без наличия естественной или искусственной вентиляции.
- Рабочее место с компьютером и оргтехникой должно размещаться на расстоянии не меньше 1м от стены, от стены с оконными проемами - на расстоянии не менее 1,5 м.
- Угол наклона экрана монитора или ноутбука по отношению к вертикали должен составлять 10-15 градусов, а расстояние до экрана - 500-600 мм.
- Угол зрения к центру экрана должен быть прямым и составлять 90 градусов.
- Для защиты от прямых солнечных лучей должны предусматриваться солнцезащитные устройства (пленка с металлизированным покрытием, регулируемые жалюзи с вертикальными панелями и др.).
- Освещение должно быть смешанным (естественным и искусственным).

- В помещении кабинета и на рабочем месте необходимо поддерживать чистоту и порядок, проводить систематическое проветривание.
- Обо всех выявленных во время работы неисправностях оборудования необходимо доложить руководителю, в случае поломки необходимо остановить работу до устранения аварийных обстоятельств. При обнаружении возможной опасности предупредить окружающих и немедленно сообщить руководителю; содержать в чистоте рабочее место и не загромождать его посторонними предметами.
- О несчастном случае очевидец, работник, который его обнаружил, или сам потерпевший должны доложить непосредственно руководителю учреждения и принять меры по оказанию медицинской помощи.

Требования безопасности перед началом работы с компьютером (ноутбуком) и другой оргтехникой.

- Осмотреть и убедиться в исправности оборудования, электропроводки. В случае обнаружения неисправностей к работе не приступать. Сообщить об этом руководителю и только после устранения неполадок и его разрешения приступить к работе.
- Проверить освещение рабочего места, при необходимости принять меры к его нормализации.
- Проверить наличие и надёжность защитного заземления оборудования.
- Проверить состояние электрического шнура и вилки.
- Проверить исправность выключателей и других органов управления персональным компьютером и оргтехники.
- При выявлении любых неисправностей, компьютер и оргтехнику не включать и немедленно поставить в известность директора школы об этом.
- Тщательно проветрить помещение с персональным компьютером и оргтехникой, убедиться, что микроклимат в помещении находится в
- допустимых пределах: температура воздуха в холодный период года - 22-24°С, в теплый период года - 23-25° С, относительная влажность воздуха — 40-60%.
- Включить монитор и проверить стабильность и четкость изображения на экране, убедиться в отсутствии запаха дыма от компьютера и оргтехники.

Требования безопасности во время работы с компьютером, ноутбуком, принтером, ксероксом, сканером, плазменной панелью, LCD-экраном и другой оргтехникой.

Включать и выключать компьютер, ноутбук и любую оргтехнику только выключателями, запрещается проводить отключение вытаскиванием вилки из розетки.

Запрещается снимать защитные устройства с оборудования и работать без них.

Не допускать к компьютеру и оргтехнике посторонних лиц, которые не участвуют в работе.

Запрещается перемещать и переносить системный блок, монитор, принтер, любое оборудование, которое находится под напряжением.

Запрещается во время работы пить какие-либо напитки, принимать пищу.

Запрещается любое физическое вмешательство в устройство компьютера, принтера, сканера, ксерокса во время их работы.

Запрещается оставлять включенное оборудование без присмотра.

Запрещается класть предметы на компьютерное оборудование, мониторы, экраны и оргтехнику.

Строго выполнять общие требования по электробезопасности и пожарной безопасности.

При работе на ксероксе и принтере во избежание поражения электротоком при устранении блокировки бумаги отключайте аппараты от сети. Отключайте оборудование от сети при длительном простое.

Самостоятельно разбирать и проводить ремонт электронной и электронно-механической части компьютера, периферийных устройств, оргтехники категорически запрещается. Эти работы может выполнять только специалист или инженер по техническому обслуживанию компьютерной техники.

Суммарное время непосредственной работы с персональным компьютером и другой оргтехникой в течение рабочего дня должно быть не более 6 часов, для педагогов — не более 4 часов в день.

Продолжительность непрерывной работы с персональным компьютером и другой оргтехникой без регламентированного перерыва не должна превышать 2-х часов. Через каждый час работы следует делать регламентированный перерыв продолжительностью 15 мин.

Во время регламентированных перерывов с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития познотонического утомления следует выполнять комплексы упражнений для глаз или организовывать физкультурные паузы.

Компьютер, любые его периферийные устройства, оргтехнику необходимо использовать в строгом соответствии с эксплуатационной документацией к ним.

Требования техники безопасности и безопасности жизнедеятельности в аварийных ситуациях при работе с компьютером и другой оргтехникой

Если на металлических частях оборудования обнаружено напряжение (ощущение тока), заземляющий провод оборван - отключить оборудование немедленно, доложить о неисправности.

При прекращении подачи электроэнергии отключить оборудование.

При появлении непривычного звука, запаха палёного, непроизвольного отключения компьютера и оргтехники немедленно остановите работу и поставьте об этом в известность руководителя.

При возникновении возгорания немедленно отключить оборудование, обесточить электросеть за исключением осветительной сети, сообщить о пожаре всем работающим и приступить к тушению очага загорания имеющимися средствами пожаротушения.

При несчастном случае необходимо в первую очередь освободить пострадавшего от травмирующего фактора, обратиться в медпункт, сохранить по возможности место травмирования в том состоянии, в котором оно было на момент травмирования. При освобождении пострадавшего от действия электротока следите за тем, чтобы самому не оказаться в контакте с токоведущей частью и под напряжением.

## 3.3. Охрана окружающей среды. Защита от энергетических воздействий. Защита от вибраций

Если в организации рабочего места оператора ПК допускается несоответствие параметров мебели антропометрическим характеристикам человека (функциональная высота рабочей поверхности стола и сидения, неудобные углы сгибания и наклона, неудачное размещение дисплея и клавиатуры и т. д.), то это вызывает необходимость поддержания вынужденной рабочей позы и может привести к нарушениям в костно-мышечной и периферической нервной системе. Длительный дискомфорт в условиях недостаточной физической активности может вызывать развитие общего утомления, снижения работоспособности, боли в области шеи, спины, поясницы. У операторов часто диагностируются заболевания опорно-двигательного аппарата и периферической нервной системы: невриты, радикулиты, остеохондроз и др.

Главной частью профилактических мероприятий в эргономике - является правильная посадка.

Высота стола должна быть приблизительно 72-75 см над уровнем пола (для человека среднего роста).

Высота стула должна быть такова, чтобы ноги стояли на полу "полной ступней".

Бедра должны быть расположены параллельно полу, а спина должна быть прямой, и отклонена немого назад.

Голова должна быть слегка наклонена вперёд. Спина, плечи и руки - расслаблены.

Расположение рук относительно стола должно быть таким, что больше половины длины предплечий упирались на стол. Руки лежат на клавиатуре согнутые в локтях под углом 90-120°.

Локти не должны висеть, ими следует опираться на подлокотники кресла. При этом подлокотники не подпирают локти и не заставляют подниматься плечи, тем самым, ущемляя шейные позвонки.

Клавиатура обязательно должна располагаться ниже локтя. Если это правило не соблюдается, то крови мышцам будет постоянно недоставать, и вы это ощутите по их быстрой усталости.

Положение кистей рук наиболее комфортно, когда угол подъема клавиатуры относительно рабочей поверхности стола составляет от 2 до 15 градусов. Такое положение обеспечит вам нормальную циркуляцию крови в кистях рук.

Кроме того, комфорт и безопасность зависят еще и от формы "мышки" - надо выбирать такую, чтобы она соответствовала вашей ладони. Если вы работаете на компьютере в разных местах, например, дома и в офисе, то постарайтесь, чтобы на том и другом рабочих местах "мышки" были одинаковые. Это избавит вас от необходимости адаптироваться, переходя на новое место.

При работе с мышкой кисть должна быть на одной прямой линии с предплечьем, для чего используется специальный коврик с подвижной опорой на колесиках или подушечкой.

Негативные последствия работы за монитором возникают из-за того, что:

а) наш глаз предназначен для восприятия отражённого света, а не излучаемого, как в случае с монитором (телевизором),

б) пользователю приходится вглядываться в линии и буквы на экране, что приводит к повышенному напряжению глазных мышц.

Для нормальной работы нужно поместить монитор так, чтобы глаза пользователя располагались на расстоянии, равном полутора диагоналям видимой части монитора:

- не менее 50-60 см для 15" монитора;
- не менее 60-70 см для 17" монитора;
- не менее 70-80 см для 19" монитора;
- не менее 80-100 см для 21" монитора.

Если зрение не позволяет выдерживать это расстояние, тогда уменьшите разрешение изображения и увеличьте шрифты.

Оптимальная диагональ экрана для работ с текстовыми документами - 15"-17" с разрешением 1024x768. Для графических работ необходим монитор 19"-21" при разрешении 1280x1024 и выше. Для игр рекомендуется 17"-19". Мониторы больших диагоналей приобретать не рекомендуется, т.к. от работы за слишком крупными мониторами, по словам пользователей, "глаза становятся квадратными".

От большого монитора необходимо сидеть дальше, чем от маленького. И в итоге угловая площадь монитора остается такой же. Но сфокусировать глаз на мелком изображении, находящемся в 1-1.5 метрах от глаза становится труднее, что ведет к перенапряжению зрительного аппарата. Чем крупнее объект на экране монитора, тем меньше утомляемость. Поэтому компьютерные игры с их рисованными фигурами утомляют меньше, чем цифры и буквы.

По высоте монитор надо располагать так, чтобы центр экрана был ниже уровня глаз. Часто мониторы располагаются так, что его центр расположен выше уровня глаз. Это не совсем правильное решение, и даже вредное, так как для шеи наиболее естественное положение - прямое, а никак не запрокидывание назад.

Монитор должен находиться прямо впереди посредине стола. Абсолютно неприемлемо расположение монитора на углу стола, когда пользователь сидит к нему вполоборота. Не стоит использовать такое решение, если вы используете компьютер больше чем 15-20 минут в день.

Плоскость экрана надо повернуть так, чтобы от верхнего и нижнего края до глаз было примерно одинаковое расстояние.

Экран монитора должен быть абсолютно чистым. Периодически и при необходимости протирайте его специальными салфетками.

Усталость от работы с монитором тем меньше, чем ниже яркость экрана и чем крупнее объекты на экране. Установите минимальную яркость, при которой можно без напряжения различать символы на экране. Учтите, что лучше увеличить шрифт или изображение, чем пододвинуться поближе к экрану или увеличить яркость. Современные операционные системы имеют для этого специальные средства. Шрифты на экране можно масштабировать, задавать минимальные размеры элементов рисунков и прочее.

На основании Федерального закона "О санитарно-эпидемиологическом благополучии населения" от 30 марта 1999 г. N 52-ФЗ разработаны и с 30 июня 2003 г. введены в действие обязательные санитарно-эпидемиологические правила и нормативы -

СанПиН 2.2.2/2.4.1340-03 "Гигиенические требования к персональным электронно-вычислительным машинам и организации работы".

На сегодняшний день это основной нормативный документ по безопасной работе на компьютере. Он введен в действие взамен СанПиН 2.2.2.542-96, действовавшего с 1997 года.

СанПиН 2.2.2/2.4.1340-03 в частности содержит конкретные санитарно-гигиенические требования к микроклимату в помещениях, где эксплуатируются ПЭВМ. Его требования сведены в таблицу.

Таблица 2

<table>
<tr><td rowspan="3">Освещение должно быть равномерным во всей комнате, особенно в рабочей зоне. Одинаково плохи и темный угол за шкафом, и непрерывно освещаемое солнцем пространство возле окна.</td><td colspan="2">Нормы освещенности</td></tr>
<tr><td>Экрана</td><td>100-250 лк</td></tr>
<tr><td>Стола</td><td>300-500 лк</td></tr>
<tr><td colspan="3">Яркость экрана должна быть не менее35 кд/м2;</td></tr>
<tr><td rowspan="4">Необходимо учитывать такие факторы, как температура, влажность и наличие пыли. Компьютеру необходимы те же условия, что и обычному человеку: температура от 21 до 25° С, умеренная влажность и отсутствие пыли.</td><td colspan="2">Микроклимат</td></tr>
<tr><td>Температура</td><td>21-25 °С</td></tr>
<tr><td>Относительная влажность</td><td>40-60 %</td></tr>
<tr><td>Скорость движения воздуха</td><td>0.1 м/с</td></tr>
</table>

| Уровни аэроионов | Число ионов в 1 см3 воздуха | |
|---|---|---|
| | п+ | п- |
| Минимально необходимый | 400 | 600 |
| Оптимальный | 1 500-3 000 | 3 000-5 000 |
| Максимально допустимый | 50 000 | 50 000 |

Далее описаны наиболее комфортные условия, в которых обеспечивается соблюдение теплового баланса температуры тела человека. При повышении температуры воздуха в помещении кровеносные сосуды расширяются, происходит повышение притока

крови к поверхности тела и теплоотдача в окружающую среду возрастает. При понижении температуры окружающей среды кровеносные сосуды сужаются и приток крови к поверхности тела, соответственно, замедляется, и теплоотдача уменьшается.

Влажность воздуха оказывает влияние на терморегуляцию организма: высокая влажность (более чем 85%) затрудняет терморегуляцию вследствие снижения испарения пота, а слишком низкая (менее 20%) - вызывает пересыхание слизистой оболочки дыхательных путей.

Движение воздуха оказывает большое влияние на самочувствие человека. Оно способствует увеличению теплоотдачи организма человека. В зимнее время года скорость движения воздуха не должна превышать 0,2-0,5 м/с, а летом - 0,2-1 м/с.

Чрезвычайно важно поддерживать оптимальную влажность в помещении. Чем выше влажность, тем сильнее ослабляется влияние электростатических полей. С другой стороны, чрезмерная влажность отрицательным образом сказывается на самочувствии людей.

Ввиду довольно жестких требований к микроклимату в компьютерных помещениях необходимо оборудовать механическую вентиляцию или кондиционеры. Но не следует забывать, что кондиционеры работают на рециркуляции и свежий воздух от их работы не поступает. Кроме того, они являются источниками шума, вредное воздействие которого описано в соответствующей главе. Поэтому мощная система приточно-вытяжной вентиляции предпочтительнее. Если вентиляция слабая, тогда рекомендуется проветривание помещение (желательно со сквозняком) несколько раз в день. Это способствует восстановлению химического состава воздуха, удалению пыли, выравниванию влажности.

Хорошо регулируют влажность в помещении комнатные растения, открытые аквариумы, офисные фонтанчики и увлажнители воздуха.

# 3. Технико-экономическое обоснование

## Заключение

Важно подчеркнуть, что в современном мире кадровая служба на предприятии играет очень важную роль. Она должна быть координатором и организатором всей работы с кадрами, кадровой политики и любых других мероприятий по работе с кадрами.

Создание автоматизированного рабочего места специалиста по кадрам на фирме позволит сократить время на обработку информации; произойдет сокращение затрат на обработку информации; уменьшатся затраты времени на поиск необходимой информации; улучшится качества контроля и учета обрабатываемой информации; повысится эффективность работы не только кадровика, но и остальных подразделений фирмы.

В данной дипломной работе разработан программный продукт для автоматизации работы отдела кадров, который отвечает поставленным изначально требованиям. В данном разделе необходимо рассмотреть соответствие требованиям задания и конечного результата. Еще раз напомним эти требования.

- Простой интерфейс;
- Минимальное количество окон;
- Модальное отображение подчиненных окон;
- Наличие главной формы приложения;
- Невозможность прямого редактирования элементов списка;
- Защита данных от несанкционированного доступа;
- Возможность регистрации;
- Автоматическое структурирование файлов списков.

Программа имеет очень простой интерфейс, интуитивно понятный, его элементы логично расположены и не требуют каких-либо дополнительных пояснений. Сами окна сгруппированы по определенному признаку таким образом, что даже совершенно неопытный пользователь имеет минимальные шансы на ошибку.

Модальное отображение подчиненных окон, как было уже более подробно пояснено в технологической части, устраняет возможную путаницу в окнах и выстраивает четкую их иерархию, значительно облегчая задачу пользователя.

Наличие главной формы обеспечивает четкое понимание основных функций программы и легкий доступ к ним.

Элементы списков редактировать напрямую нельзя, это исключает случайные записи в базе данных, представляющих собой структурированные файлы текстового формата.

Вход в программу с процедурой авторизации предусмотрен для исключения возможности пользования программой случайными людьми. Осуществлять регистрацию нового пользователя имеет право только пользователь с правами администратора.

Автоматическое структурирование файлов текстового формата необходимо для правильной расстановки данных и их корректного отображения в визуальных компонентах программы. Также это оставляет возможность для расширения функционала программы, например, подключения таких компонентов, как Table.

Расчёт экономической эффективности проекта показал, что предполагаемая прибыль от применения выполненной разработки программы для автоматизации работы отдела может составить …… руб.

Таким образом, выдвинутые задачи дипломного проектирования решены, поставленные цели достигнуты, а разработанная программа для автоматизации работы отдела кадров актуальна, целесообразна и экономически эффективна.

**Список использованной литературы**

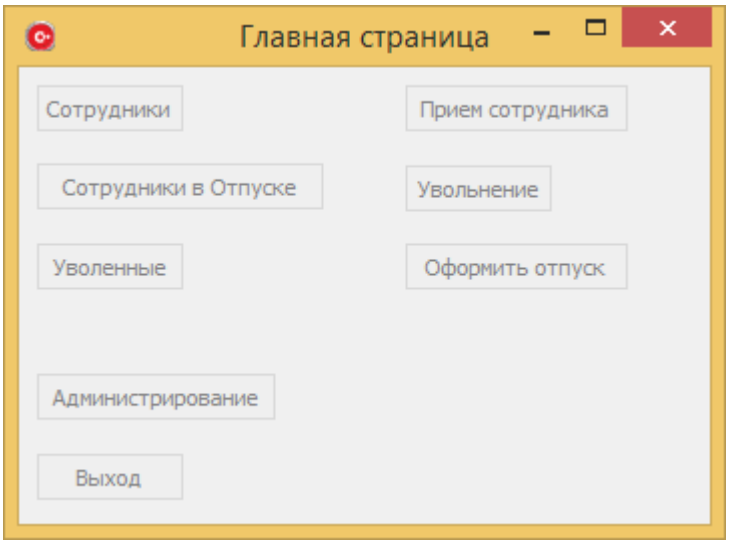

1. Блэк Рекс. Ключевые процессы тестирования: планирование, подготовка, проведение, совершенствование: Пер. с англ. – М.: «Лори», 2010. – 566 с.:ил.
2. Винтонива, Н.И. Информационные технологии управления персоналом: учебное пособие. – Владивосток : Изд-во ВГУЭС, 2010. – 136 с.
3. Дэвис, Стефан, Р. C++ для "чайников", 4-е издание. : Пер. с англ. : — М. : Издательский дом '"Вильямс", 2013. — 336 с. : ил. : Парал. тит. англ.
4. Информационные технологии в управлении персоналом : учебник и практикум для прикладного бакалавриата / Ю. Д. Романова, Т. А. Винтова, П. Е. Коваль, П. А. Музычкин. — М. : Издательство Юрайт, 2014. — 291 с. — Серия : Бакалавр. Прикладной курс.
5. Коплиен Дж. Мультипарадигменное проектирование для C++. Библиотека программиста. — СПб.: Питер, 2010. —235 с: ил.
6. Купер А., Рейман Р., Кронин Д. Алан Купер об интерфейсе. Основы проектирования взаимодействия. – Пер. с англ. – СПб.: Символ'Плюс, 2010. – 688 с., ил.
7. Литвиненко Н. А. Технология программирования на C++. Win32 API-приложения. — СПб.: БХВ-Петербург, 2010. — 288 с.: ил. — (Учебное пособие)
8. Медведев В.И. Особенности объектно-ориентированного программирования на C++/CLI, C# и Java. 2-е изд., испр. и доп. - Казань: РИЦ «Школа», 2010.-444 с.: ил. Гриффитс Д. Изучаем программирование на C; пер. с англ. /Дэвид Гриффитс, Дон Гриффитс. – М.:Эксмо, 2013. – 624 с. : ил.
9. Мюссер, Дэвид Р., Дердж, Жилмер Дж., Сейни, Атул. C++ и STL: справочное руководство, 2-е изд. (серия C++ in Depth).: Пер. с англ. — М.: 000 "И.Д. Вильяме", 2010. — 432 с.: ил. — Парал. тит. англ.
10. Пахомов Б.И. C/C++ и MS Visual C++ 2010 для начинающих. – СПб.: БХВ-Петербург, 2011. – 736 с.: ил.
11. Страуструп, Бьярне. Программирование: принципы и практика использования C++. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2011. — 1248 с.: ил. — Парал. тит. англ.
12. Хортон А. Visual C++ 2010: полный курс.: пер. с англ. – М.: «ООО И.Д. Вильямс», 2011. – 1216 с.: ил. Парал. тит. англ.
13. Шлее М. Qt4.5. Профессиональное программирование на C++. – СПб: БХВ-Петербург, 2010. – 896 с.:ил.

14. Эванс Эрик. Предметно-ориентированное программирование (DDD): структуризация сложных программных систем: Пер. с англ. – М.: «ООО И.Д.Вильямс», 2011. – 448с.:ил.

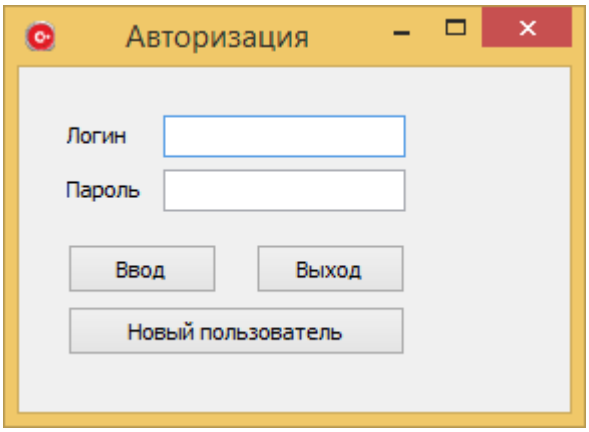
Интернет-ресурсы:

15. http://blogkadrovika.ru/besplatnaya-programma-dlya-otdela-kadrov/
16. http://citforum.ru/programming/application/program/
17. http://citforum.ru/programming/c.shtml
18. http://codeinlife.ru
19. http://cpp.com.ru
20. http://cppstudio.com
21. http://protoplex.ru/soft_group/41.html
22. http://purecodecpp.com
23. http://software-testing.ru/forum/index.php?/forum/78-literatura-po-testirovaniiu-po/
24. http://www.c-cpp.ru
25. http://www.consultingstandard.ru/it_hr.htm
26. http://www.cyberguru.ru/visual-cpp/visual-cpp-beginners.html
27. http://www.programmersclub.ru/01/
28. http://www.protesting.ru/testing/
29. https://code-live.ru/tag/cpp-manual/
30. https://ru.wikibooks.org/wiki/Си%2B%2B

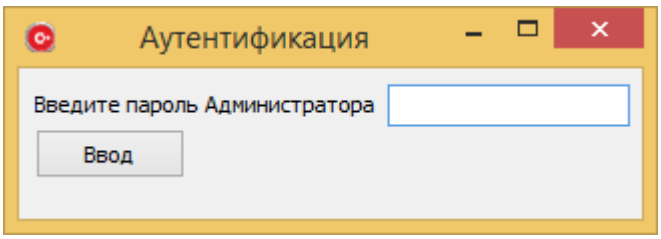
**Приложение А**
**Руководство пользователя**




Работа в программе начинается с ввода логина-пароля пользователя. Работать могут только авторизованные пользователи.

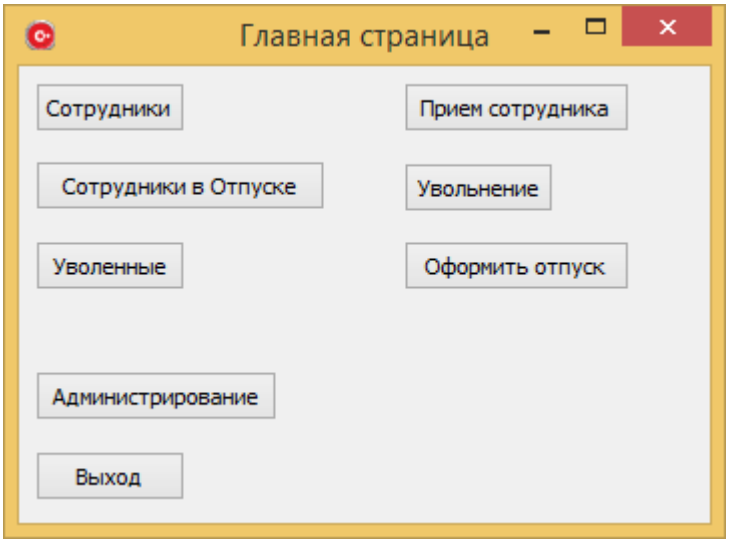



Если данных для входа нет, необходимо пройти процедуру регистрации. С этой целью необходимо пройти следующую процедуру:

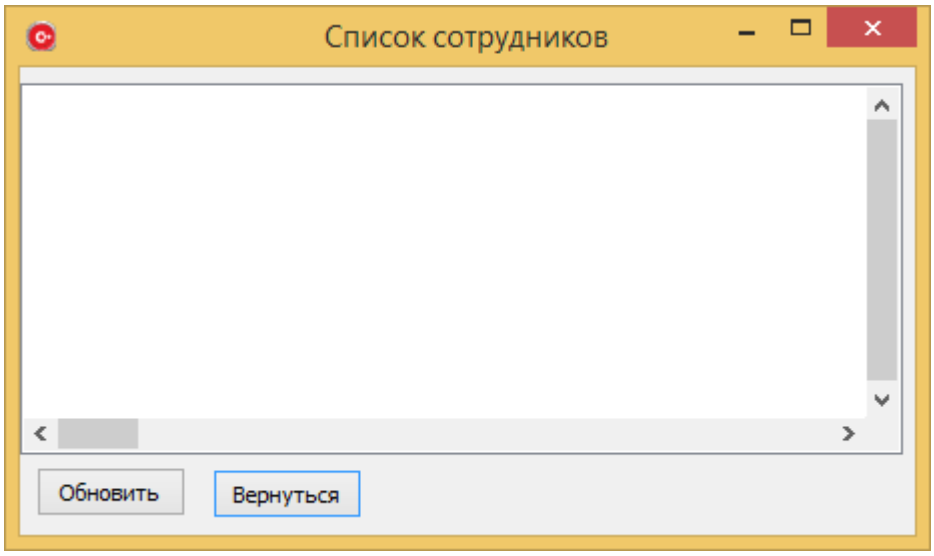



Для выполнения процедуры регистрации необходим административный пароль. Он известен только системному администратору или иному лицу, выполняющему инсталляцию программы и имеющий доступ к открытому исходному коду.

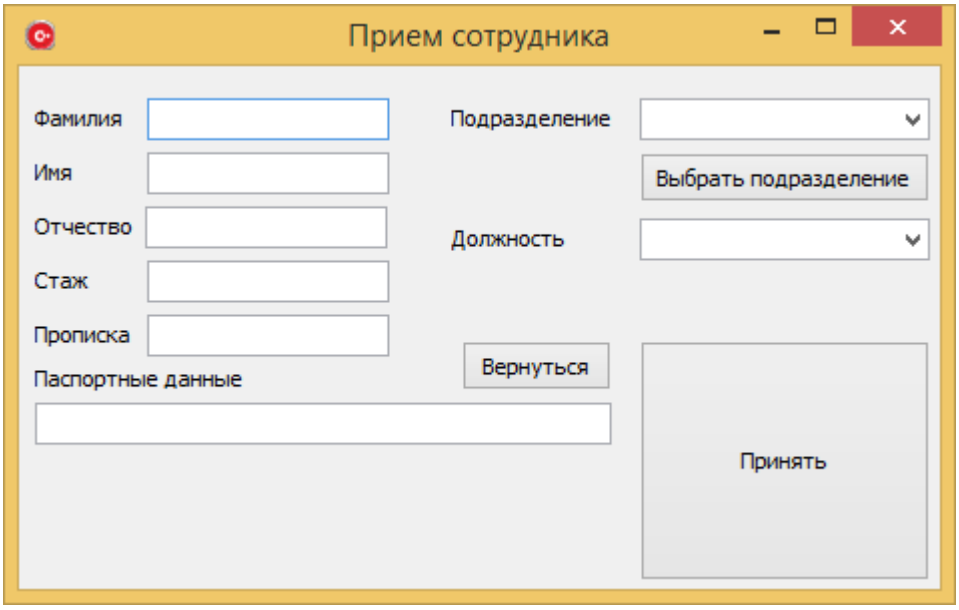
После прохождения процедуры авторизации пользователь получает доступ к главному окну:




Если требуется просто просмотреть список сотрудников, достаточно нажать кнопку «Сотрудники».



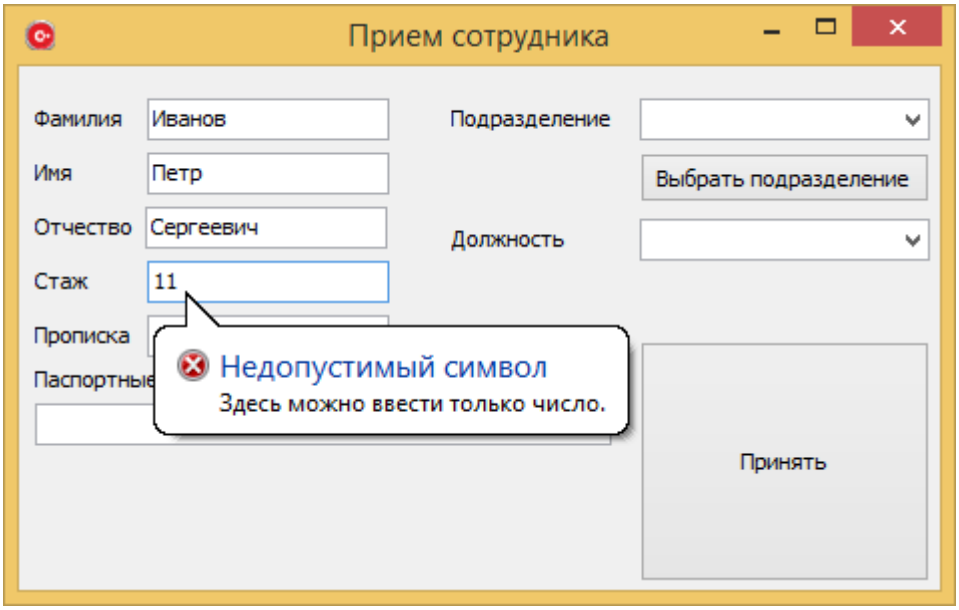

Так выглядит чистая база данных сотрудников в начале работы программы. Стоит заметить, что сотрудника в этом окне добавить невозможно, соответстсвующие инструменты отсутствуют. Это сделано для того, чтобы все данные заносились только с помощью определенной формы по строго заданным правилам.




Чтобы добавить сотрудника в базу, его нужно сначала принять на работу. После нажатия кнопки «Прием сотрудника» появляется следующее окно.

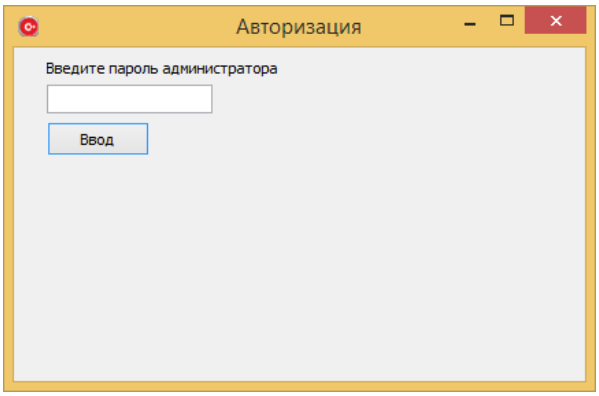
При заполнении необходимо соблюдать следующие правила:

1) Инициалы сотрудника не должны содержать в себе никаких символов, кроме букв
2) В поле «Стаж» не допускается ввод иных символов, кроме цифр (максимальное число – 99)
3) Подразделение выбирается из выпадающего списка
4) Должность выбирается из выпадающего списка

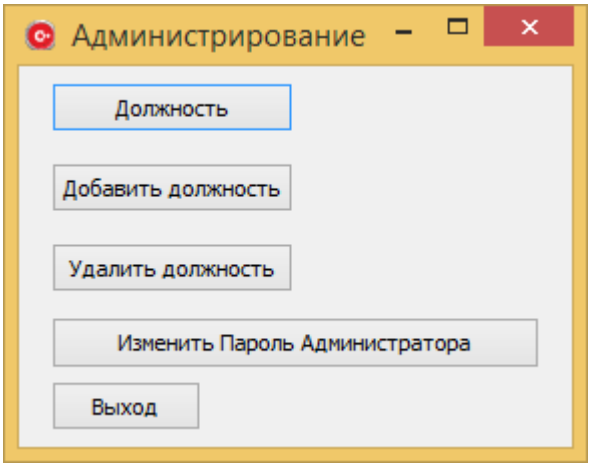



Если изменения, внесенные в окне, правильны, то следует нажать кнопку «Принять». При нажатии кнопки «Вернуться» изменения не сохраняются, окно закрывается и пользователь возвращается в главное окно.

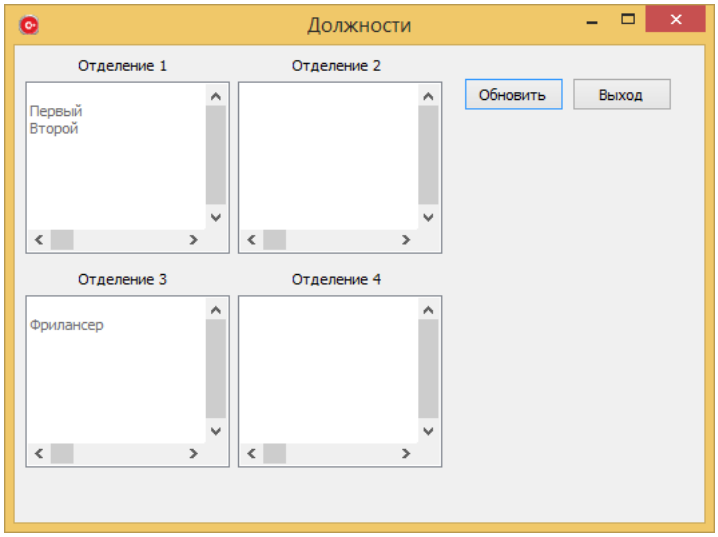
Начальные данные частично заполняются с помощью администратора. Например, сотрудник отдела кадров не может самостоятельно редактировать списки подразделений и соответствующих им должностей. Это можно сделать только после ввода административного пароля. Чтобы отредактировать эти списки, необходимо нажать кнопку «Администрирование».




После ввода пароля появляется следующее окно.

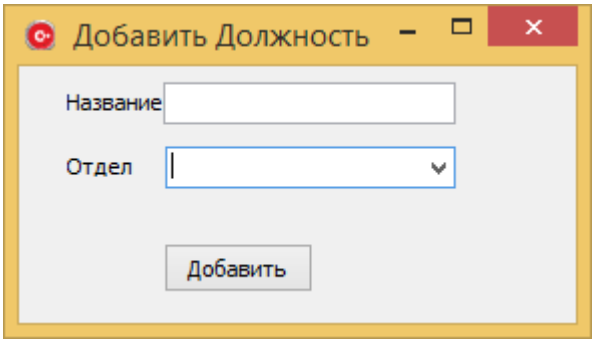


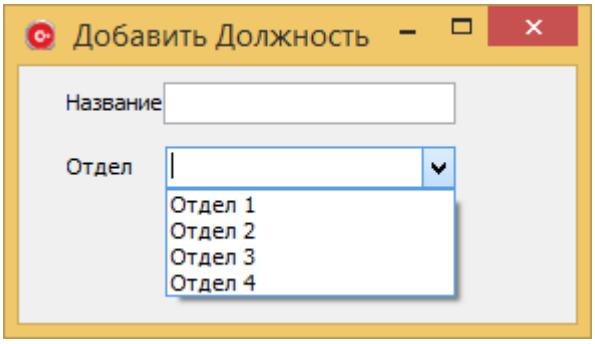

Так выглядит список должностей при нажатии кнопки «Должность».



Как хорошо видно на данном скриншоте, профессии отсортированы согласно подразделениям. Их «миграция» недопустима.

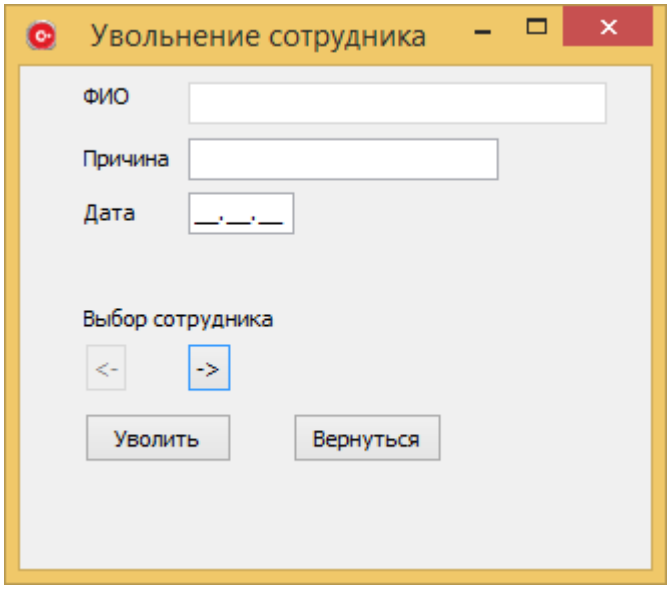
При нажатии кнопки «Добавить должность» появляется следующее окно:




Объект «Новый Отдел» в этом окне создать нельзя. Это действие выполняется исключительно системным администратором из соображений целесообразности: новые отделы в фирмах появляются крайне редко, и нет необходимости усложнять программный код.

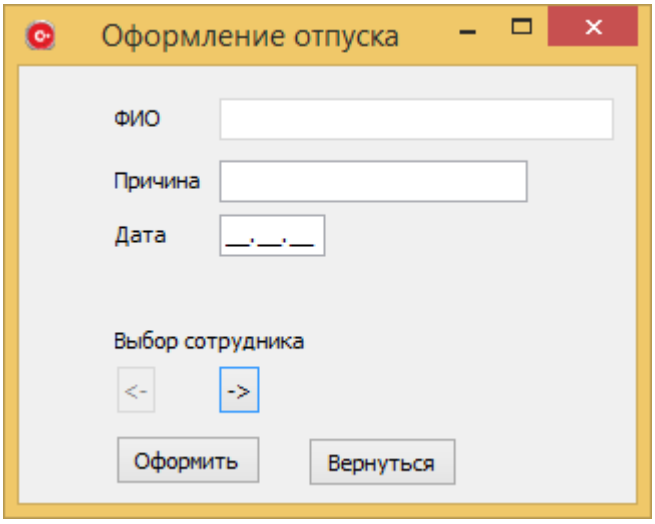
При необходимости сотрудника можно уволить или отправить в отпуск. Чтобы это сделать, необходимо нажать на соответствующие кнопки главной формы приложения.



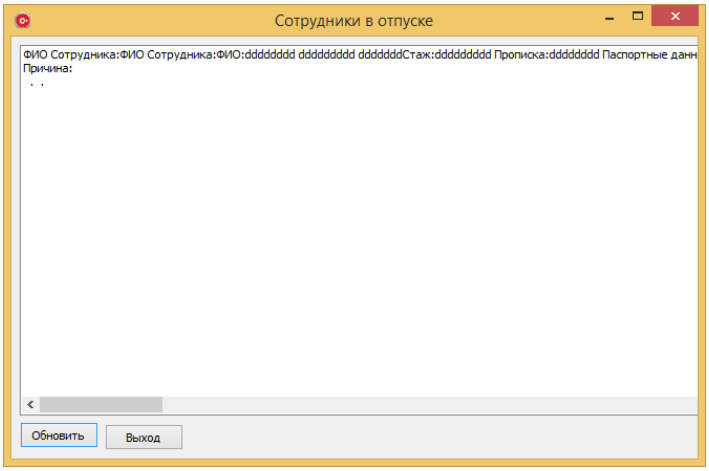

Из данного скриншота хорошо видно, что невозможно уволить сотрудника из пустой БД – поле «ФИО» недоступно. В обязательном порядке нужно указать причину увольнения –

номер статьи ТК и причину расторжения трудового договора. Также необходимо указать дату увольнения. В поле предусмотрена маска ввода, которое не позволит ввести вместо даты другие символы.

Аналогичным образом оформляется отпуск.



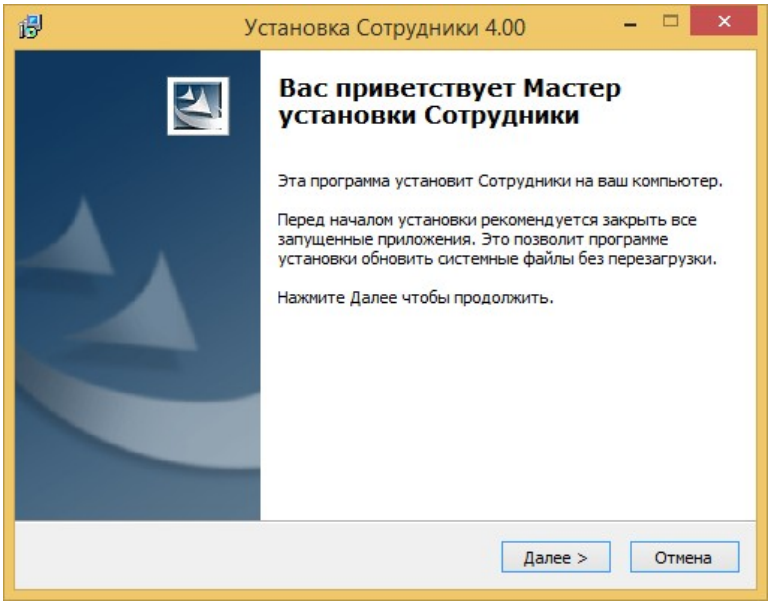

Одной из особенностей БД является возможность различных выборок. В данном случае одним из признаков выборки является наличие возможности просмотра сотрудников, находящихся в отпуске, и тех, кто когда-либо работал в организации.



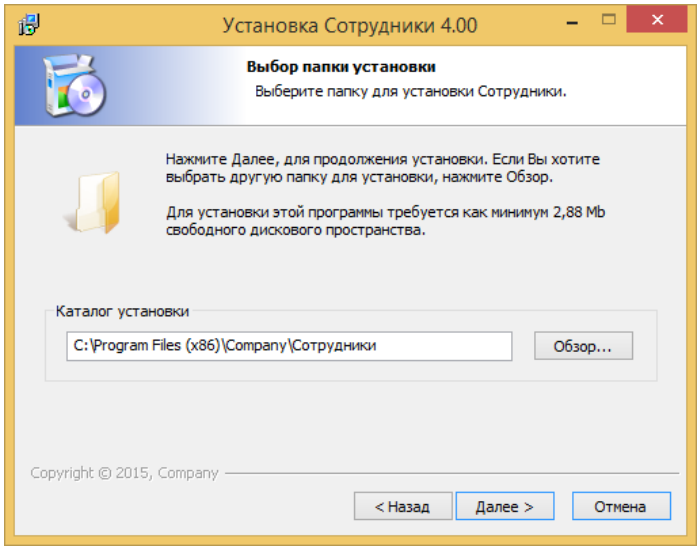

Форма для просмотра всех сотрудников, уволенных из организации, выглядит аналогично.

Для установки программы необходимо выполнить следующие действия:

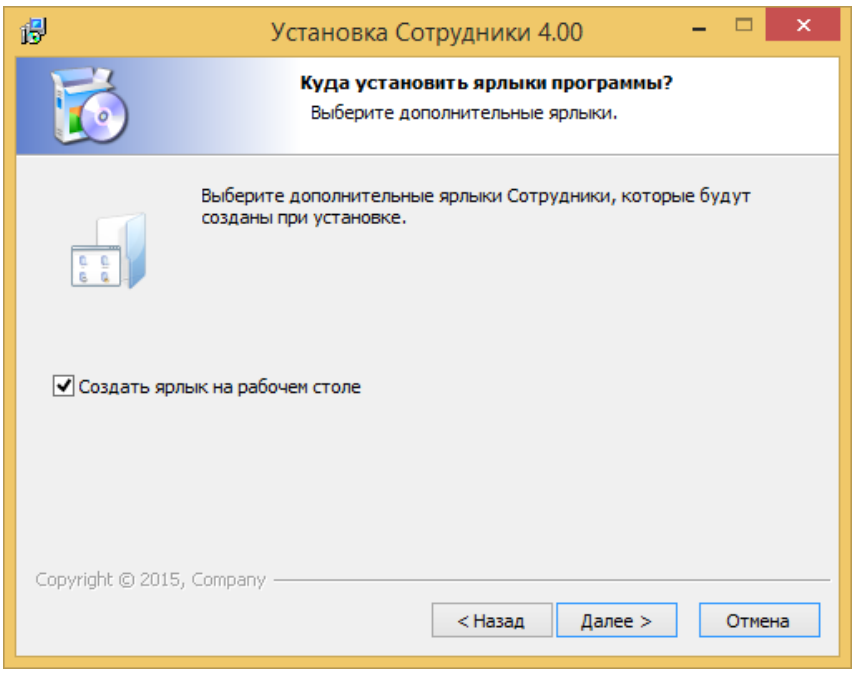
1) Запустить установочный файл sotrudniki.exe
2) Утвердительно ответить на вопрос о разрешении установки программы на компьютере
3) Нажать кнопку «Далее» в первом окне Мастера установки:



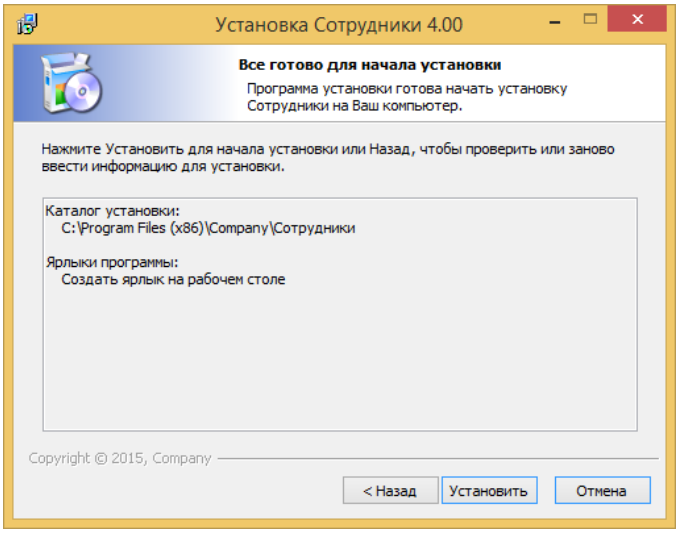

4) Указать каталог установки программы (желательно не менять параметры по умолчанию):



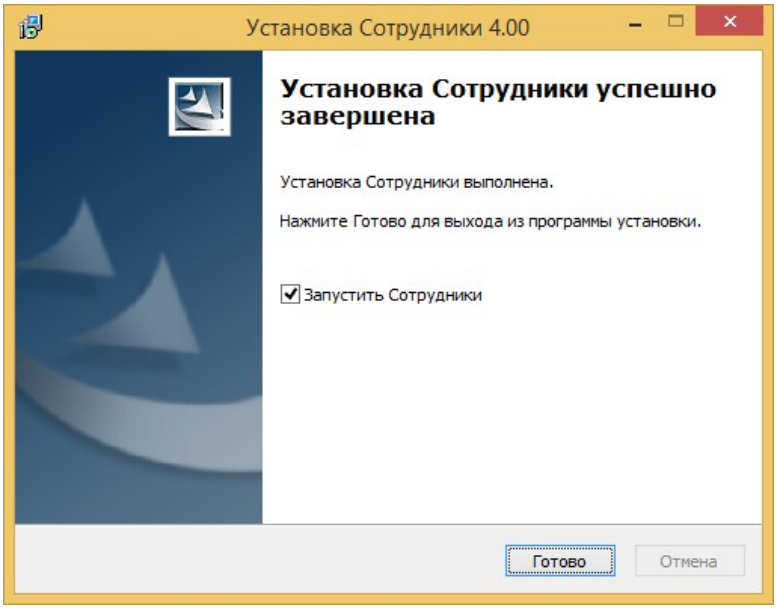

5) Выбрать дополнительные параметры установки (создать ярлык программы на рабочем столе):




6) Убедиться в том, что все параметры установлены правильно, и нажать кнопку «Установить»:

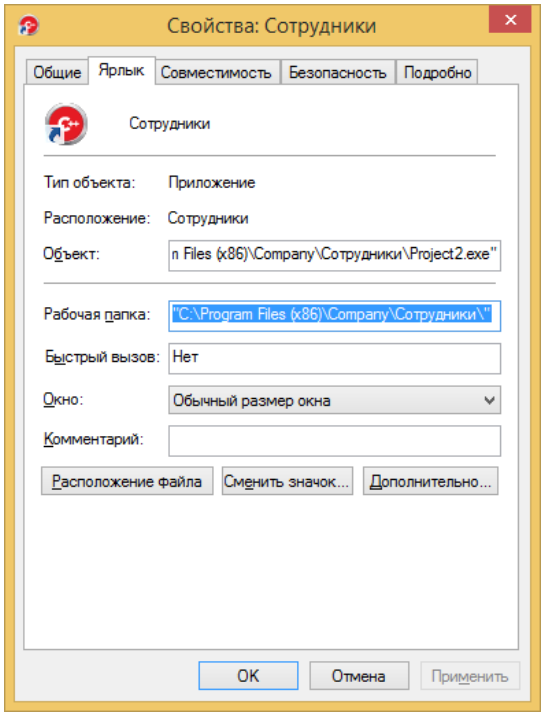



7) В заключительном окне Мастера установки будет предложено запустить только что установленную программу. Если это не требуется, снять соответствующий флажок

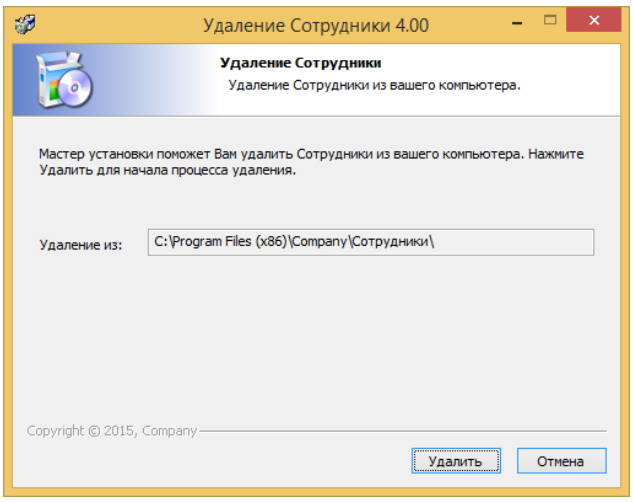



Деинсталлляция (удаление) программы производится следующим образом:

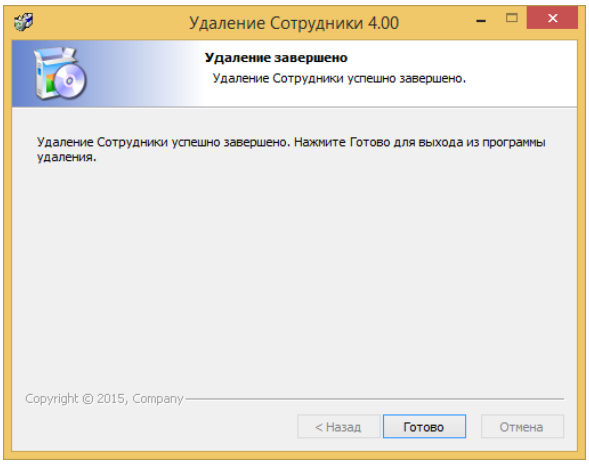
1) Нужно войти в папку "C:\Program Files (x86)\Company\Сотрудники\". Чтобы не искать ее, щелкнуть правой кнопкой на ярлыке программы «Сотрудники» на рабочем столе, нажать кнопку «Расположение файла»:




2) Щелкнуть дважды на файле Uninstall.exe. появится окно Мастера удаления программы:




3) После нажатия кнопки «Удалить» программа будет удалена с компьютера и появится соответствующее оповещение.



**Приложение Б**
**Руководство программиста**

В данном разделе приводятся исходные некоторых компонентов программы с комментариями. Представлены не все файлы кода, т.к. многие из них очень похожи друг на друга и при необходимости программист может найти аналог компонента, не описанного в настоящем руководстве.

Содержимое Unit1.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
int a;
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
if (a!=0) {a=a-1;Edit1->Text = Memo1->Lines->Strings[a];Button2->Enabled=True;Button3->Enabled=True;} else {Button1->Enabled=False;};
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
a=0;
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
if (Edit1->Text!="")
{
Edit3->Text="ФИО Сотрудника:";
Edit4->Text="Причина:";
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Uwl.txt");
Edit3->Text=Edit3->Text+Edit1->Text;
Edit4->Text=Edit4->Text+Edit2->Text;
l1->Add(Edit3->Text);
l1->Add(Edit4->Text);
l1->Add(MaskEdit1->Text);
l1->SaveToFile("C:/Pred/Uwl.txt")  ;
if (a!=0) {
Memo1->Lines->Delete(a);
Memo2->Lines->Delete(a+1);
Memo2->Lines->Delete(a+1);
Edit1->Text="";}
else{
Memo1->Lines->Delete(a);
Memo2->Lines->Delete(a);
Memo2->Lines->Delete(a);
Edit1->Text="";};
TStringList* l2=new TStringList;
l2->AddStrings(Memo1->Lines);
l2->SaveToFile("C:/Pred/Sotr.txt");
Button3->Enabled=False;
}
else
{
MessageBox(NULL,L"Вы совершили ошибку",L"Ошибка",MB_OK );}
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
if (Edit1->Text!="") {a=a+1;Edit1->Text = Memo1->Lines->Strings[a];Button1->Enabled=True;Button3->Enabled=True;}
else {Button2->Enabled = False;} ;
}
//---------------------------------------------------------------------------
```

Файл Unit2.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
#include "Unit3.h"
#include "Unit6.h"
#include "Unit4.h"
#include "Unit7.h"
#include "Unit5.h"
#include "Unit8.h"
#include "Unit9.h"
#include "Unit17.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
int a;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm2::Button1Click(TObject *Sender)
{
Form3->Show();     //Открываем форму Сотрудники
Form3->ListBox1->Items->Clear();
TStringList* l1=new TStringList;//Объявление файловой переменной
l1->LoadFromFile("C:/Pred/Sotr.txt"); //Загрузка данных из файла
Form3->ListBox1->Items->AddStrings(l1);
Form3->ListBox1->Perform(LB_SETHORIZONTALEXTENT,4024,0);                    //создание горизонтальных полос прокрутки
ShowScrollBar(Form3->ListBox1->Handle, SB_VERT, True);  //создание вертикальных полос прокрутки

}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button2Click(TObject *Sender)
{
a=0;            //Открываем форму Приема
Form6->Show();
Form6->ListBox1->Items->Clear();
Form6->ComboBox2->Items->Clear();
 TStringList* l2=new TStringList;
l2->LoadFromFile("C:/Pred/Sotr.txt");
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Otd1.txt");


Form6->ComboBox2->Items->AddStrings(l1);
Form6->ListBox1->Items->AddStrings(l2);
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button3Click(TObject *Sender)
{
a=0;
```

```
Form4->Show();  //Открываем форму Увольнения
Form4->ListBox1->Items->Clear();
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Sotr.txt");
Form4->ListBox1->Items->AddStrings(l1);

Form4->Edit1->Text = Form4->ListBox1->Items->Strings[0];
Form4->Edit5->Text=StrToInt(Form4->ListBox1->Items->Count-1);



Form4->Button1->Enabled=False;
Form4->Button2->Enabled=True;

}
//---------------------------------------------------------------------------


void __fastcall TForm2::Button4Click(TObject *Sender)
{
Form7->Show();      //Открываем форму Список уволенных
Form7->ListBox1->Items->Clear(); //отчистка объекта листбокс1
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Uwl.txt");
Form7->ListBox1->Items->AddStrings(l1);
Form7->ListBox1->Perform(LB_SETHORIZONTALEXTENT,4000,0);
ShowScrollBar(Form7->ListBox1->Handle, SB_VERT, True);

}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button6Click(TObject *Sender)
{
a=0;          //Открываем форму Оформления в отпуск
Form5->Show();
```

```
Form5->ListBox1->Items->Clear();
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Sotr.txt");
Form5->ListBox1->Items->AddStrings(l1);
Form5->Edit1->Text = Form5->ListBox1->Items->Strings[0];
Form5->Edit5->Text=StrToInt(Form5->ListBox1->Items->Count-1);
Form5->Button1->Enabled=False;
Form5->Button2->Enabled=True;
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button5Click(TObject *Sender)
{
Form8->Show();        //Открываем форму Список сотрудников в отпуске
Form8->ListBox1->Items->Clear();
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Otp.txt");
Form8->ListBox1->Items->AddStrings(l1);
Form8->ListBox1->Perform(LB_SETHORIZONTALEXTENT,4000,0);
ShowScrollBar(Form8->ListBox1->Handle, SB_VERT, True);

}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button7Click(TObject *Sender)
{
Form9->Show();       //Открываем форму Администрирования
Form9->Edit1->Text="";
}
//---------------------------------------------------------------------------


void __fastcall TForm2::FormClick(TObject *Sender)
{
if (Button1->Tag==0) {
```

```
Form17->Show();
}
}
//---------------------------------------------------------------------------



void __fastcall TForm2::Button8Click(TObject *Sender)
{
Form2->Close();  //Закрываем форму
```

Файл Unit3.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button2Click(TObject *Sender)//Обновляем данные в листбоксе из файла
{
ListBox1->Items->Clear();
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Sotr.txt");
ListBox1->Items->AddStrings(l1);
```

```
ShowScrollBar(ListBox1->Handle, SB_VERT, True);
}
//---------------------------------------------------------------------------



void __fastcall TForm3::Button1Click(TObject *Sender)
{
Form3->Close();
}
//---------------------------------------------------------------------------
```

Файл Unit4.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
int a;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Button1Click(TObject *Sender)
{
if (a!=0) {a=a-1;Edit1->Text = ListBox1->Items->Strings[a];Button2->Enabled=True;Button3-
>Enabled=True;}
 else {Button1->Enabled=False;Button2->Enabled=True;}; //выбор сотрудника(в лево)
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Button2Click(TObject *Sender)
{
if (a!=StrToInt(Edit5->Text)) {a=a+1;Edit1->Text = ListBox1->Items->Strings[a];Button1->Enabled=True;Button3->Enabled=True;}
else {Button2->Enabled = False;Button1->Enabled=True;} ; //выбор сотрудника (стрелка вправо)
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Button3Click(TObject *Sender) //данное действие загрузит данные из файла, удалит из него нужную строку и занесет отчет в другой файл
{
if (Edit1->Text!="" || Edit2->Text!="" || Edit3->Text!="")//проверка на заполненость
{
Edit3->Text="ФИО Сотрудника:";
Edit4->Text="Причина:";
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Uwl.txt"); //загрузка данных из файла в файловую переменную
Edit3->Text=Edit3->Text+Edit1->Text;
Edit4->Text=Edit4->Text+Edit2->Text;
l1->Add(Edit3->Text); //построчное добавление данных в переменную
l1->Add(Edit4->Text);
l1->Add(MaskEdit1->Text);
l1->SaveToFile("C:/Pred/Uwl.txt") ; //сохранение данных из строковой переменной в файл
ListBox1->Items->Delete(a);//удаление выбраной строки из объекта листбокс
Edit1->Text="";
TStringList* l2=new TStringList;
l2->AddStrings(ListBox1->Items);
l2->SaveToFile("C:/Pred/Sotr.txt");
Button3->Enabled=False; // блокировка кнопки необходима дл защиты от случайных действий
ListBox1->Items->Clear();
Form4->Close(); //форма закрывается для защиты от случайных действий
}
```

```
else
{
MessageBox(NULL,L"Вы совершили ошибку",L"Ошибка",MB_OK );}//данное сообщение появится если оставить строки пустыми
}
//---------------------------------------------------------------------------
void __fastcall TForm4::FormCreate(TObject *Sender)
{
a=0;
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button4Click(TObject *Sender)
{
Form4->Close(); //закрывает форму
}
//---------------------------------------------------------------------------
```

Файл Unit5.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit5.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
int a;
//---------------------------------------------------------------------------
__fastcall TForm5::TForm5(TComponent* Owner)
        : TForm(Owner)
{
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm5::Button1Click(TObject *Sender)
{
if (a!=0) {a=a-1;Edit1->Text = ListBox1->Items->Strings[a];Button2->Enabled=True;Button3->Enabled=True;}
 else {Button1->Enabled=False;Button2->Enabled=True;}; //выбор сотрудника(в лево)
}
//---------------------------------------------------------------------------
void __fastcall TForm5::Button2Click(TObject *Sender)
{
if (a!=StrToInt(Edit5->Text)) {a=a+1;Edit1->Text = ListBox1->Items->Strings[a];Button1->Enabled=True;Button3->Enabled=True;}
else {Button2->Enabled = False;Button1->Enabled=True;}  ;  //выбор сотрудника(в право)
}
//---------------------------------------------------------------------------
void __fastcall TForm5::Button3Click(TObject *Sender)
{
if (Edit1->Text!="") //проверка на заполненость
{
Edit3->Text="ФИО Сотрудника:";
Edit4->Text="Причина:";
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Otp.txt");
Edit3->Text=Edit3->Text+Edit1->Text;
Edit4->Text=Edit4->Text+Edit2->Text;
l1->Add(Edit3->Text);
l1->Add(Edit4->Text);
l1->Add(MaskEdit1->Text);
l1->SaveToFile("C:/Pred/Otp.txt")  ;
Button3->Enabled=False;
ListBox1->Items->Clear();
Form5->Close();
}
else
```

```
{
MessageBox(NULL,L"Вы совершили ошибку",L"Ошибка",MB_OK );}
}
//---------------------------------------------------------------------------
void __fastcall TForm5::FormCreate(TObject *Sender)
{
a=0;
}
//---------------------------------------------------------------------------

void __fastcall TForm5::Button4Click(TObject *Sender)
{
Form5->Close();

}
//---------------------------------------------------------------------------
```

Файл Unit6.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit6.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;
//---------------------------------------------------------------------------
__fastcall TForm6::TForm6(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm6::Button1Click(TObject *Sender)
{
if (Edit1->Text!="" || Edit2->Text!="" || Edit3->Text!=""  //проверка на заполненость
&& Edit4->Text!="" ||
Edit5->Text!="" ) {
Edit7->Text="ФИО:"+Edit1->Text+" "+Edit2->Text+" "+Edit3->Text;
Edit8->Text="Стаж:"+Edit4->Text+" Прописка:"+Edit5->Text+" Паспортные данные:"
+Edit6->Text+" Подразделение:"+ComboBox1->Text+" Должность:"+ComboBox2->Text;
Edit8->Text=Edit7->Text+"          "+Edit8->Text;
ListBox1->Items->Add(Edit8->Text) ;//добавление строки в обЪект листбокс из объекта едит8
TStringList* l1=new TStringList; //объявление строковой переменной
l1->AddStrings(ListBox1->Items); //загрузка данных в строковую переменную из обЪеккта листбокс
l1->SaveToFile("C:/Pred/Sotr.txt"); //сохранение данных из строковой переменной в файл
Edit1->Text="";
Edit2->Text="";
Edit3->Text="";
Edit4->Text="";
Edit5->Text="";
Edit6->Text="";
ComboBox1->Text="";
ComboBox2->Text="";
ComboBox2->Items->Clear();
Form6->Close();
} else {MessageBox(NULL,L"Вы оставили одно из полей пустым",L"Ошибка",MB_OK );};  //данное сообщение появится если оставить поля пустыми
}
//---------------------------------------------------------------------------
void __fastcall TForm6::Button2Click(TObject *Sender)
{
Form6->Close();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm6::Button3Click(TObject *Sender)
{

TStringList* l1=new TStringList;  //подгружаем список должностей из файлов
if (ComboBox1->Text!="") {
ComboBox2->Items->Clear();
  if (ComboBox1->Text=="Отделение 1") { //после проверки в строковую переменную добавятся  данные из выбранного файла
 l1->LoadFromFile("C:/Pred/Otd1.txt");
 ComboBox2->Items->AddStrings(l1) ;}
  if (ComboBox1->Text=="Отделение 2") {
 l1->LoadFromFile("C:/Pred/Otd2.txt");
 ComboBox2->Items->AddStrings(l1); }
  if (ComboBox1->Text=="Отделение 3") {
 l1->LoadFromFile("C:/Pred/Otd3.txt");
 ComboBox2->Items->AddStrings(l1); }
  if (ComboBox1->Text=="Отделение 4") {
 l1->LoadFromFile("C:/Pred/Otd4.txt");
 ComboBox2->Items->AddStrings(l1); }
}
}
//---------------------------------------------------------------------------


void __fastcall TForm6::Edit1KeyPress(TObject *Sender, System::WideChar &Key)
{
  AnsiString ForbiddenChars = "0123456789,.*/-+=\|<>?:;%№""!)(*&^%$#@!~";  //запрет на ввод выбраных символов (цифры и знаки)
if(ForbiddenChars.Pos(Key)) Key = 0;
 if (Key==0){MessageBox(NULL,L"Вы ввели запрещенный символ",L"Ошибка",MB_OK );};

}
//---------------------------------------------------------------------------
```

Файл Unit10.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit10.h"
#include "Unit11.h"
#include "Unit12.h"
#include "Unit13.h"
#include "Unit14.h"
#include "Unit15.h"
#include "Unit16.h"
#include "Unit20.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm10 *Form10;

//---------------------------------------------------------------------------
__fastcall TForm10::TForm10(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm10::Button1Click(TObject *Sender)
{
Form11->Show();
Form11->ListBox1->Items->Clear();
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Otd.txt");
Form11->ListBox1->Items->AddStrings(l1);
Form11->ListBox1->Perform(LB_SETHORIZONTALEXTENT,4000,0);
ShowScrollBar(Form11->ListBox1->Handle, SB_VERT, True);
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm10::Button2Click(TObject *Sender)
{
Form12->Show();
Form12->ListBox1->Items->Clear();
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Otd.txt");
Form12->ListBox1->Items->AddStrings(l1);

}
//---------------------------------------------------------------------------
void __fastcall TForm10::Button3Click(TObject *Sender)
{

Form13->Show();
Form13->ListBox1->Items->Clear();
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Otd.txt");
Form13->ListBox1->Items->AddStrings(l1);
Form13->Edit1->Text = Form13->ListBox1->Items->Strings[0];
Form13->Edit5->Text=StrToInt(Form13->ListBox1->Items->Count-1);
Form13->Button1->Enabled=False;
Form13->Button2->Enabled=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm10::Button4Click(TObject *Sender)  //открываем список должностей в разных отделах
{
Form14->Show();
Form14->ListBox1->Items->Clear();
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Otd1.txt");
Form14->ListBox1->Items->AddStrings(l1);
Form14->ListBox1->Perform(LB_SETHORIZONTALEXTENT,4000,0);
ShowScrollBar(Form14->ListBox1->Handle, SB_VERT, True);
```

```
Form14->ListBox2->Items->Clear();
TStringList* l2=new TStringList;
l2->LoadFromFile("C:/Pred/Otd2.txt");
Form14->ListBox2->Items->AddStrings(l2);
Form14->ListBox2->Perform(LB_SETHORIZONTALEXTENT,4000,0);
ShowScrollBar(Form14->ListBox2->Handle, SB_VERT, True);

Form14->ListBox3->Items->Clear();
TStringList* l3=new TStringList;
l3->LoadFromFile("C:/Pred/Otd3.txt");
Form14->ListBox3->Items->AddStrings(l3);
Form14->ListBox3->Perform(LB_SETHORIZONTALEXTENT,4000,0);
ShowScrollBar(Form14->ListBox3->Handle, SB_VERT, True);

Form14->ListBox4->Items->Clear();
TStringList* l4=new TStringList;
l4->LoadFromFile("C:/Pred/Otd4.txt");
Form14->ListBox4->Items->AddStrings(l4);
Form14->ListBox4->Perform(LB_SETHORIZONTALEXTENT,4000,0);
ShowScrollBar(Form14->ListBox4->Handle, SB_VERT, True);

}
//---------------------------------------------------------------------------

void __fastcall TForm10::Button5Click(TObject *Sender)
{
Form15->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm10::Button6Click(TObject *Sender)
{
Form16->Show();
Form16->ListBox1->Items->Clear();
```

```
TStringList* l1=new TStringList;
l1->LoadFromFile("C:/Pred/Otd1.txt");
Form16->ListBox1->Items->AddStrings(l1);
Form16->Edit1->Text = Form16->ListBox1->Items->Strings[0];
Form16->Edit5->Text=StrToInt(Form16->ListBox1->Items->Count-1);
Form13->Button1->Enabled=False;
Form13->Button2->Enabled=True;
}
//---------------------------------------------------------------------------

void __fastcall TForm10::Button7Click(TObject *Sender)
{
Form20->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm10::Button8Click(TObject *Sender)
{
Form10->Close();
}
//---------------------------------------------------------------------------
```