УДК 342.3:331.546

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ОБУЧЕНИЯ И КОНТРОЛЯ ЗНАНИЙ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ И ОХРАНЕ ТРУДА

канд. техн. наук А.В. СПИРИДОНОВ, канд. техн. наук В.Б. ХАЛИЛ
(Полоцкий государственный университет)

Представлена методика тестового контроля знаний сотрудников нефтехимических производств. Предложено программное обеспечение, а также рассмотрены вопросы использования вычислительной техники для обучения технике безопасности и охране труда персонала промышленных предприятий и в учебном процессе по дисциплине «Информационные технологии в охране труда» ИПК УО «ПГУ». Применение компьютерного тестирования для оперативного контроля уровня знаний, рассмотренного в данной работе, обладает рядом преимуществ перед традиционными методами контроля. Представлен анализ поставленных задач. Рассмотрена организация программной структуры, логическая и физическая структура базы данных, проектирование и взаимодействие классов, редактор, создание интерфейса серверной и клиентской части, а также приложение для просмотра, особенности реализации программного обеспечения и организация сетевого взаимодействия. Показано, что данная разработка позволяет автоматизировать аспект подготовки кадров, способствуя лучшей организованности и результативности работников, а именно более безопасной и эффективной работе.

С интенсивным развитием информационных технологий и проникновением их в системы управления технологическими и бизнес-процессами предприятий и организаций всё более значимой становится роль программных продуктов в сфере обучения и проведения тестирования кадров предприятия. Аттестация персонала становится общепризнанным способом оценки результативности труда и, как следствие, важным средством для управления и организационного развития [1].

Во многих компаниях регулярно проводятся аттестационные интервью, собеседования или тестирование, контроль знаний по технике безопасности и охране труда, причем для разных профессий критерии оценивания могут существенно отличаться. Особенно такая оценка актуальна на предприятиях, где присутствует повышенный риск для здоровья сотрудника, а работа сопряжена со многими вредными факторами. К таким предприятиям можно причислить предприятия нефтехимического комплекса [2].

Показатели, по которым оцениваются работники, называют критериями оценки. Сюда относятся, в частности, качество выполняемой работы, ее количество и ценностная оценка результатов. Важным здесь является то, что оцениваться должны не личности, а результативность работы.

Наряду с внедрением инструментов, контролирующих качество знаний работников предприятия, важны и другие решения, влияющие на знания сотрудников. Ведь качество знаний – это основная цель контроля. Таким инструментом может быть средство обучения персонала, так как этап обучения всегда должен предшествовать этапу контроля знаний [3].

В условиях текущей экономической ситуации в Республике Беларусь, характеризующейся увеличением производства и привлечением новых сотрудников, подготовка и обучение производственных кадров является важной задачей. Данная разработка позволяет автоматизировать аспект подготовки кадров, тем самым способствуя лучшей организованности и результативности работников, и как следствие – сделает работу более безопасной и эффективной. Одной из причин несчастных случаев, чрезвычайных ситуаций и аварий на предприятиях являются неправильные действия персонала. Это связано с недостаточной подготовкой специалистов и работников. На многих предприятиях сегодня обучение персонала проводится самостоятельно, а на результаты контроля знаний часто оказывают влияние субъективные факторы.

Анализ поставленных задач. Анализ существующих программ контроля знаний показал их некоторые недостатки:

- общедоступность базы данных (отсутствие защиты), хранящейся на общем ресурсе;
- не все программы имеют возможность оперативного редактирования и пополнения базы данных в связи с изменяющимися требованиями, нормативными документами и т.п.;
- при прохождении обучения пользователь не имеет возможности просмотра литературы и графической информации, относящейся к данному вопросу.

Организация программной структуры. Контроль знаний – двусторонний процесс. При проведении тестирования всегда выделяются два связанных основных субъекта – экзаменатор и экзаменуемый. Эти две сущности, несмотря на связь, при взаимодействии друг с другом имеют разные права, полномочия и, следовательно, разные доступные модели поведения [4].

Экзамен предполагает возможность одновременного прохождения экзамена несколькими сотрудниками. Решение, позволяющее достичь этой цели, – использование сетевого взаимодействия, когда эк-

заменуемые, находясь на разных компьютерах, проходят тест у одного экзаменатора. Данная модель имеет несколько преимуществ:

- возможность проводить тестирование нескольких сотрудников одновременно, что позволяет экономить время экзаменатора;
- возможность следить за экзаменом на одном рабочем месте;
- упрощается хранение и обновление данных о сотрудниках, графиках проведения аттестационных экзаменов.

Итак, для разработки модели проведения тестирования в локальной сети необходимы как минимум две программы: программа-клиент и программа-сервер. Программа-клиент осуществляет связь с сервером, подключаясь к нему и получая все необходимые данные. Программа-сервер позволяет наблюдать за процессом экзамена.

Основой для работы программ служит база данных. Она содержит в себе всю необходимую информацию для проведения обучения и контроля знаний (рис. 1).

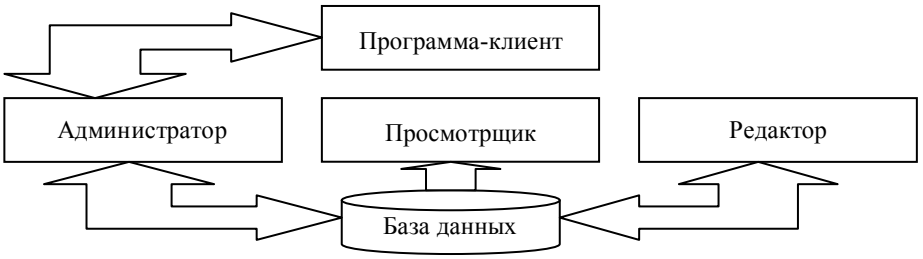

Рис. 1. Организация программной структуры

Так как программа-сервер работает с информацией из базы данных только в рамках пересылки необходимых данных экзаменуемому (клиенту), возникает потребность еще в одной программе – *редакторе*. Данная программа в *авторизованном режиме* позволяет экзаменатору добавлять и редактировать вопросы, изменять информацию о сотрудниках, оперировать данными на уровне отделов и т.п.

База данных, кроме непосредственного участия в экзамене в качестве экзаменуемого, необходима сотрудникам, так как в процессе обучения сотрудники могут изучать различные литературные источники, а следовательно и обращаться к базе данных. Еще одна функция программного обеспечения для обучения и контроля знаний – определение даты экзамена. Программа-редактор содержит все необходимые сведения, однако не может служить для этой цели по следующим причинам:

- режим доступа в программе-редакторе не исключает злонамеренного изменения данных в целях фальсификации результатов экзамена, т.е. режим редактирования не приемлем для обычного сотрудника;
- доступный интерфейс – рядовой работник должен иметь дело с предельно простой программой, что не обеспечивается программой-редактором.

Итак, решением может быть создание еще одного приложения, на которое будут возложены вышеперечисленные функции. Таким образом, выделен минимальный набор программ, составляющий программное обеспечение для обучения и контроля знаний по технике безопасности и охране труда:

- программа-редактор;
- администратор экзаменов, обеспечивающий наблюдение за ходом экзамена;
- программа-клиент, с помощью которой экзаменуемый соединяется с сервером и проходит процесс тестирования и обучения;
- программа для просмотра предстоящих экзаменов и просмотра соответствующей литературы.

Логическая и физическая структура базы данных. База данных хранит в себе следующую информацию [2, 3]:

- *данные о сотрудниках*. Сюда включается персональная информация о сотруднике, профессия, а также информация, позволяющая отслеживать график проведения тестирования;
- *данные о профессиях*. Каждая профессия имеет характеристики, влияющие на вид проведения тестирования. Например, профессии, предполагающие большую ответственность имеют более высокий процент правильных ответов для успешной сдачи экзамена;
- *экзаменационные вопросы* (текст вопроса; варианты ответов; правильный ответ; должность, к которой относится вопрос; литература по данному вопросу; графический файл к вопросу);
- *литература* (название файла с литературой; должность, к которой относится литература; краткое описание или комментарий).

Структура базы данных представлена на рисунке 2.

Внутренняя структура базы данных отражает все тонкости ее функционирования и, несмотря на схожесть с логической структурой, имеет ряд нюансов, зависящих от сложности реализации.

Для организации записей используем такой элемент языка С, как структура struct. Количество структур – это количество выделенных сущностей, так как вся информация о них хранится на носителе. То есть создадим следующие структуры:

- Post – структура для хранения данных о должностях;
- Staff – структура для хранения данных о сотрудниках;
- Lit – структура для хранения данных о литературе;
- Quest – структура для хранения данных об экзаменационном вопросе.

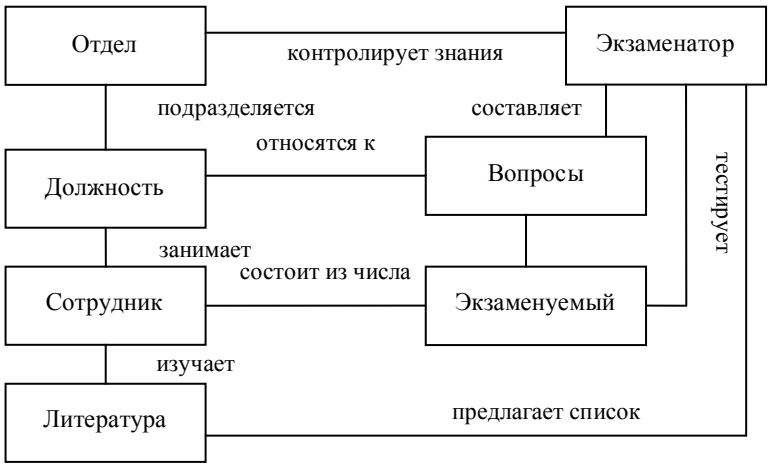

Рис. 2. Структура базы данных

Определим оптимальное количество ответов на вопрос. Считается, что оптимальное количество ответов – два или три. Таким образом, для записи Quest введем три поля – максимально необходимое количество, – отведенные под ответ. Такая схема хранения данных наиболее удобна, когда все данные сгруппированы в одном файле. Чтобы соответствовать этому критерию, будем хранить все данные о должностях, сотрудниках, литературе и вопросах в одном файле. Это позволяет получить сразу несколько преимуществ:

- решается вопрос об отображении структуры предприятия, так как данные о каждом отделе группируются в отдельном файле;
- появляется возможность легко манипулировать сгруппированными данными (резервное копирование, архивирование и т.п.).

Возникает вопрос, как сохранить данные для того, чтобы в будущем их легко было извлечь, при этом уменьшив избыточную информацию.

Наиболее простым решением было бы закрепление за каждой структурой определенного количества байт, т.е. каждая структура, независимо от хранимого количества информации, должна иметь одинаковый размер. Такой подход наиболее прост в реализации, более того, упрощает и процедуру считывания данных. Он подходит для первых трех структур: Staff, Post и Lit. Однако текст вопроса и ответов может быть разным. Если выделить слишком малый размер полей, то возникнут ограничения на длину вопроса и ответов. Еще один вариант – указание длины вопроса в базе данных. Таким образом, усложняется доступ к произвольному вопросу, зато решается проблема рационального использования ресурсов.

Проектирование и взаимодействие классов. Так как база данных является ядром программы, необходимо выделить класс, отвечающий за работу с ней. Итак, выделим класс, отвечающий за работу с базой данных. Назовем этот **класс TBase**. Атрибуты данного класса:

- список должностей (vPost);
- список сотрудников (vStaff);
- список литературы (vLit);
- список вопросов (vQuest);
- имя файла (file_name).

В данном случае база данных представляет собой информацию одного отдела или подразделения.

Определим конструктор класса, он будет иметь параметр – имя базы данных. Логика использования данного класса такова – при открытии базы данных какого-либо отдела создается экземпляр объекта TBase. При создании конструктору передается всего лишь имя базы данных. Задача объекта – при вызове метода Load найти по названию подразделения соответствующий файл и считать его данные в память.

Данный класс применяется для редактирования базы данных: добавления, удаления и изменения хранимой информации. Все методы, возвращающие какие-либо значения, имеют префикс Get, а устанав-

ливающие – Set. Данный класс изолирует физическую структуру файла от остальной части программы, что существенно упрощает работу с базой данных.

Следующий класс, выполняет аналогичные функции, однако имеет другую реализацию TServerBase. Этот класс инкапсулирует в себе список вопросов, количество вопросов на экзамене, процент ответов для успешной сдачи экзамена, логическую переменную, указывающую на факт прохождения экзамена и дату последнего экзамена. Основное назначение данного класса – работа с базой данных; позволяет решать следующие задачи:

- загружать вопросы для определенной должности;
- извлекать информацию о сотруднике;
- сохранять данные о результатах экзамена;
- формировать список отделов, должностей и сотрудников.

Сущность «Вопрос» позволяет выделить соответствующий класс. Так как в данном случае вопрос понимается как нечто более широкое (не только текст вопроса, но и ответы), то и атрибуты данного класса будут соответствующими. Также данный класс хранит тот ответ, который дал экзаменуемый; является строительным кирпичом для других классов, так как это основная минимальная единица, с которой сотрудник имеет дело на экзамене. С другой стороны, экзаменатор также использует данную сущность, составляя вопрос и ответы и предлагая их экзаменуемому. Такая двусторонняя связь предполагает использование данного класса как при проектировании клиентской, так и при проектировании серверной программы. Она же определяет наличие соответствующих методов.

При создании экземпляра **класса TQuestion** конструктору передаются все необходимые параметры. Перед тем как давать ответ на вопрос, сотрудник должен ознакомиться с вопросом, вариантами ответов и прикрепленной к вопросу картинкой, схемой, если таковая имеется.

При подведении итогов учитывается ответ, данный сотрудником и правильный ответ. Следующие методы специально служат для этой цели. Они не имеют параметров, а их возвращаемое значение – число, означающее номер ответа.

Для упрощения получения результатов вводится специальный метод, позволяющий определить, верен данный ответ или нет. Он также берет на себя функцию определять, считать ли верным ответ, если на него не дан никакой ответ. Например, если экзаменуемый досрочно прервал экзамен и не ответил на вопрос, ответом будет «нет». Данный метод не имеет параметров, а тип возвращаемого значения – логический.

Класс, отвечающий за проведение экзамена TExam. Экзамен имеет следующие характеристики: количество вопросов, процент правильных ответов для успешного прохождения, список вопросов.

Пусть классTClientExam унаследует все свойства TExam. Теперь остается добавить те специфические особенности, которые необходимы клиентской стороне. Текущий вопрос, который просматривает клиент, и ответ на текущий вопрос – основные отличия данного класса. Появляются и соответствующие методы:

- переход к следующему вопросу (NextQuestion);
- получение номера текущего вопроса (GetCurQuestionNum);
- метод, возвращающий ссылку на текущий вопрос (GetCurQuestion);
- метод, обеспечивающий сохранение ответа пользователя (AcceptAnswer).

Программа-сервер может обслуживать несколько клиентов одновременно. Поэтому возникает необходимость создания еще одного класса, содержащего атрибуты и методы для обслуживания клиента. Назовем этот класс TClient. Он содержит те атрибуты, которые можно выделить у каждого экзаменуемого, а именно: каждый экзаменуемый проходит некоторый экзамен, содержащий набор вопросов. Он также характеризуется названием отдела, должностью, именем, датой последнего экзамена, датой начала и окончания текущего экзамена. Также необходимо ввести некоторые атрибуты, отвечающие на вопрос, проходил ли сотрудник экзамен когда-либо в прошлом и закончен ли текущий экзамен.

Вся эта информация необходима для наблюдения за прохождением экзамена. От подключения до окончания экзамена клиент проходит различные стадии. Для того чтобы отобразить эти стадии на экране наблюдателя, вводится еще один дополнительный атрибут – статус.

Класс TClient использует класс TExam. При подключении создается экземпляр класса TClient, но объект класса TExam еще не инициализируется. Инициализация происходит в начале экзамена. До того как инициализирован пользователь, класс TExam не используется. В начале экзамена вызывается метод Init класса TExam. Проинициализировавшись, объект типа TExam хранит в себе список вопросов, вынесенных на экзамен. Класс TExam хранит вектор объектов типа TQuestion. При этом загружаются все вопросы для данного экзаменуемого сотрудника. После этого выбираются случайные вопросы и в класс TExam добавляются элементы класса TExam. После завершения формирования вопросов, экземпляр класса TBase уничтожается и освобождает память.

Таким образом, чтобы получить доступ к данным о подключенном сотруднике, используются методы класса TClient, а при просмотре экзаменационных вопросов данный класс подставляет интерфейс

другого класса, отвечающего за экзамен – TExam. Класс TExam в свою очередь для доступа к вопросам предоставляет интерфейс класса TQuestion. Методы данного класса используются для получения данных о вопросе, ответе пользователя, определения корректности ответа и т.д.

Редактор. Программа редактор является основой серии разрабатываемых приложений (рис. 3). В процессе эксплуатации именно эта программа запускается впервые для того, чтобы сформировать базу данных. Она осуществляет редактирование базы данных, т.е. с ее помощью базу можно пополнить данными, исправить неточности, удалить ненужные вопросы. Она также управляет структурой базы данных, отражающей структуру предприятия. То есть программа-редактор осуществляет управление каждым файлом, который по своей сути является данными по конкретному отделу.

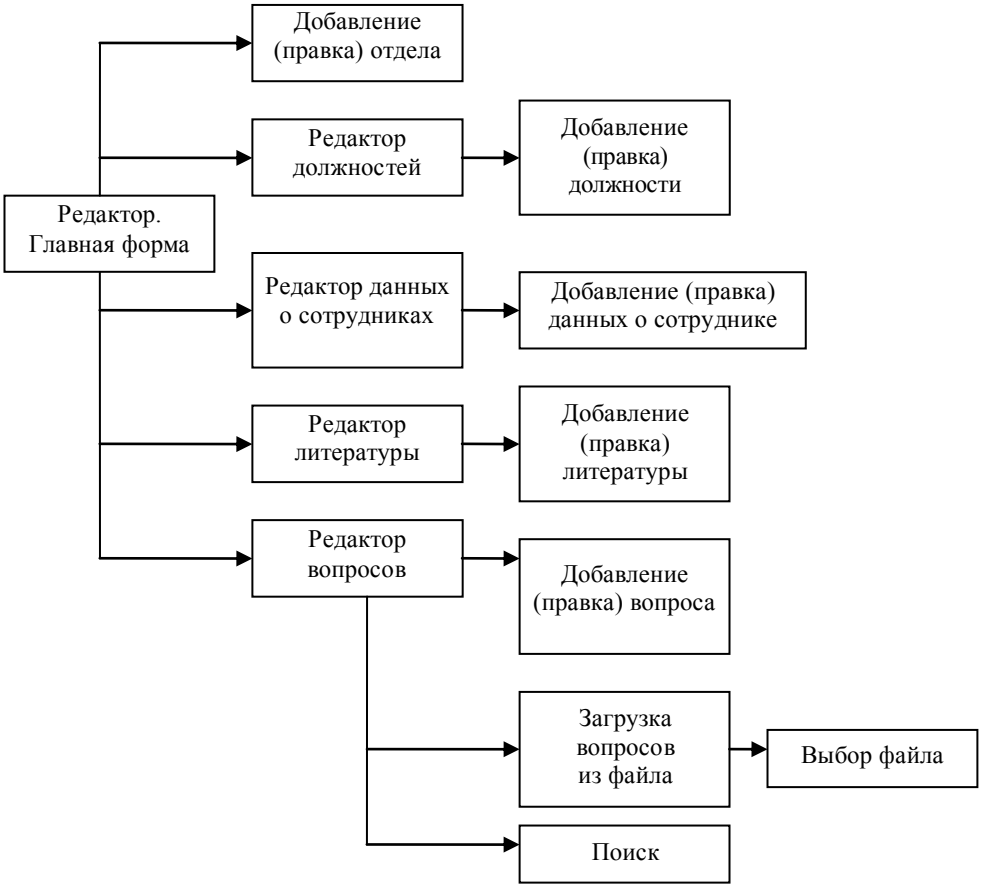

Рис. 3. Иерархическая структура форм редактора

Проектирование редактора начинается с главного окна. Окно содержит функциональную и информативную части. Функциональная часть представлена набором основных возможностей, которые предоставляет редактор:

- редактирование отделов;
- редактирование должностей;
- редактирование данных о сотрудниках;
- редактирование вопросов;
- сохранение базы данных;
- выход из программы.

Основная единица информации, с которой работает база данных в один момент времени – отдел. Таким образом, отображение текущего отдела можно вынести и визуально выделить на форме.

С помощью компонента TComboBox можно просматривать все имеющиеся отделы, а также выбирать текущий отдел. За информативную часть интерфейса отвечает компонент TStatusBar. Он имеет несколько панелей, каждая из которых содержит информацию, сколько объектов содержится в базе данных. При смене активного отдела панель обновляет свои данные. Кнопки редактирования информации об отделе находятся непосредственно на форме. Редактирование остальных составных частей базы данных будет происходить во вложенных формах, чтобы не загромождать главную форму. Наличие пиктограмм обес-

печит интуитивно понятное функциональное назначение кнопок. Сами кнопки группируются вместе по тому же признаку: кнопки вызова дополнительных редакторов – в одной колонке, кнопки редактирования информации об отделе – непосредственно под списком отделов, прочие кнопки – в другой колонке. Редактор вопросов имеет другую структуру, чем вышеперечисленные редакторы. Это обусловлено форматом хранимых данных – структурой вопросов. Функциональность редактора рассчитана на максимальное удобство пользователя. Редактор позволяет осуществлять удобную навигацию с помощью четырех кнопок: «В начало», «Назад», «Вперед», «В конец».

При работе с обширной базой данных целесообразно организовать поиск. Поиск может осуществляться по тексту вопроса или по должности. Текст предлагается вводить вручную, тогда как должности можно выбрать из списка. Существуют две кнопки поиска – одна для включения режима поиска, другая для просмотра следующего результата поиска. В обычном режиме кнопка просмотра следующего результата недоступна. По аналогии с редактором литературы, присутствует кнопка «Копировать», позволяющая не вводить вопрос повторно, если, например, требуется ввести много однотипных вопросов. Это также позволяет лишний раз не указывать должность, литературу и графический элемент к вопросу.

Важная функция редактора вопросов – загрузка вопросов из текстового файла. Если предположить, что материал, содержащий список вопросов, копился до внедрения данной программы, то ее заполнение может быть упрощено. При нажатии соответствующей кнопки возникает окно, позволяющее выбрать, должности, к которым относятся загружаемые вопросы, и литературу к ним. Присоединение данных должно производиться автоматически, исходя из выбора пользователя.

Когда пользователь выбрал эти параметры, возникает диалоговое окно, позволяющее выбрать текстовый файл. После нажатия кнопки «ОК» происходит процесс преобразования текстовых данных, и таблица заполняется. После окончания заполнения пользователь видит главное окно редактора вопросов с уже загруженными вопросами и может работать с ними так же, как если бы он вводил их вручную.

Создание интерфейса серверной и клиентской части. Главная задача – отображать подключения, адекватно реагировать на клиентские компьютеры. При этом как можно больше информации должно быть доступно для наблюдателя. Важнейшей информацией для наблюдателя-экзаменатора являются данные о подключенных пользователях. Для полной идентификации пользователя необходимы такие атрибуты, как фамилия, имя, отчество экзаменуемого, отдел и должность. После подключения экзаменуемый сотрудник может проходить разные стадии (рис. 4).

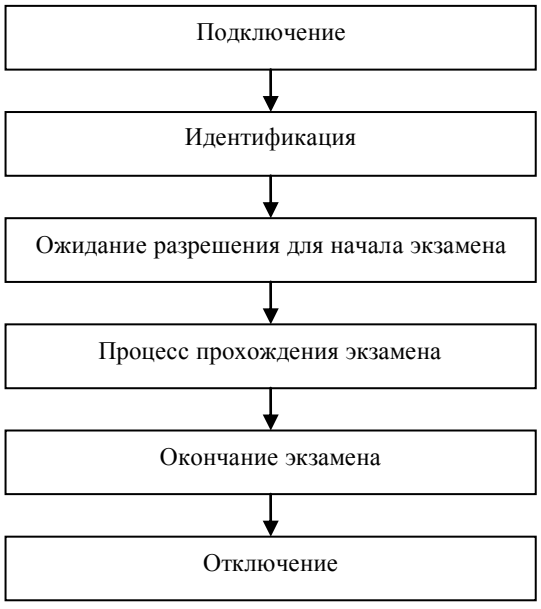

Рис. 4. Стадии экзамена для клиента

Для отображения соответствующих стадий необходимо ввести строку статуса пользователя. Наилучшим решением будет, если наблюдатель будет видеть одновременно все данные всех подключенных пользователей. Для этого может служить компонент TListView. Он позволяет каждому клиенту присвоить какое-либо графическое отображение в зависимости от текущего статуса. Помимо этого этот компонент решает задачу по отображению нескольких атрибутов пользователя, что улучшает интерфейс программы. Каждый вопрос необходимо снабдить пиктограммой, которая позволит наблюдателю увидеть, на какой вопрос пользователь ответил (или не ответил), и правильность ответа на данный вопрос. Также можно пометить правильные или неправильные ответы.

Клиентская часть предъявляет повышенные требования к простоте и наглядности интерфейса, так как зачастую пользователь имеет минимальный опыт работы на компьютере. Первое, что должен будет осуществить пользователь – это подключение к удаленному серверу. Возникает пользовательское окно, где имеется список адресов компьютеров, к которым сотрудник может подключиться. Нажав кнопку и подключившись к серверу, пользователь осуществляет дальнейшие действия.

В работе всегда происходят непредвиденные задержки и паузы, что особенно касается программ, осуществляющих сетевое взаимодействие. Для того чтобы пользователь был проинформирован о каких-либо действиях компьютера, необходимо организовать сообщение об этом.

После подключения происходит идентификация клиента. Чтобы клиенту было проще идентифицировать себя для сервера, используем каскадный выбор данных. Это означает, что сначала пользователь выбирает отдел, к которому он относится, затем должность и, наконец, свою фамилию. После выбора

всех параметров отображается информация, сколько вопросов содержит экзамен, необходимый процент правильных ответов для успешной сдачи экзамена и дата последнего экзамена. По окончании ввода собственных параметров пользователь приступает к экзамену. Он ожидает разрешения от сервера и после этого может начать экзамен. Для этого пользователь нажимает соответствующую кнопку на панели инструментов, с этого момента все остальные кнопки панели инструментов становятся доступными – экзамен начинается. Слева пользователь может видеть список вопросов, которые графически маркируются для того, чтобы пользователь знал, на какие вопросы он ответил, а на какие еще предстоит дать ответ.

Важно предоставить пользователю возможность пропускать вопросы, если ответ на них вызывает какие-либо трудности. Если к вопросу относится графическая часть, то пользователь должен это четко видеть. Для этого становится видимой сама картинка в сжатом масштабированном виде справа от текста вопроса. Возникает окно, в котором пользователь может просмотреть графическую часть вопроса. Строка состояния отображает общее количество вопросов и номер текущего вопроса, а также некоторый комментарий, который подсказывает пользователю дальнейшие действия.

При завершении программы экзаменуемый сотрудник может ознакомиться с результатами. Возникает окно, в котором графически помечены правильные и неправильные ответы. Окно имеет ту же структуру, что и в серверной части. Внизу окна отображаются данные о сотруднике, краткая статистика и результат экзамена.

Выбор режима проведения тестирования доступен пользователю в окне идентификации. Сотрудник использует переключатель для выбора режима. При выборе режима обучения сотрудник не указывает свою фамилию, а только отдел и специальность.

Приложение для просмотра. Данная программа должна обеспечивать работу следующих функций:

- ознакомление с датой проведения экзамена;
- просмотр и поиск литературы для работника определенной должности.

Каждый сотрудник, таким образом, сможет, запустив программу, сразу выбрать свой отдел, а далее просматривать необходимую ему информацию, используя соответствующую вкладку. Просматривать предстоящие экзамены можно до определенного дня, при желании пользователя он может изменить этот параметр. По умолчанию просмотр ведется до текущей даты. Если необходимо просмотреть также тех сотрудников, которые еще не проходили экзамен, или данные об этом отсутствуют в базе, ставится специальный флажок «Просматривать также с неизвестной датой экзамена». Результат просмотра – список сотрудников, также даты прохождения экзамена.

Просмотреть литературу сотрудник может на другой вкладке. Если должностей слишком много – можно выбрать свою должность и список автоматически изменится. По умолчанию выбран просмотр литературы для всех должностей. Открыть файл с литературой можно, либо нажав кнопку «Просмотр», либо дважды щелкнув по соответствующему элементу.

Особенности реализации программного обеспечения. Редактирование данных осуществляется посредством программы-редактора. Все файлы по отделам хранятся в одной папке, там же, где расположена программа. Для осуществления редактирования необходимо запустить программу. Редактор автоматически обращается к списку отделов, находящемуся в файле dep.dat. Список всех существующих отделов выводится на экран. При выборе нового отдела программа находит соответствующий отделу файл в папке Data и открывает его для чтения. Считав данные можно осуществить их просмотр и редактирование.

При совершении изменений данные меняются только в оперативной памяти компьютера, для сохранения на диск необходимо нажать соответствующую кнопку. Это позволяет отменить неправильные действия, загрузив отдел заново. Отдел имеет два параметра: название и имя файла. При желании название файла можно изменить, тогда файл автоматически переименуется.

Вопросы часто содержат литературу и графическую часть. Просмотр графики осуществляется внутренними средствами программы, тогда как для просмотра литературы используется ассоциированная с ней внешняя программа. Данное действие осуществляет API-функция ShellExecute.

Литература и картинки к вопросам находятся также в одной отдельной папке для избежания повторений. Часто бывает, что одна и та же литература используется в разных отделах. Чтобы избежать ненужного повторения файлов, использована идея одной папки для всей информации как текстовой, так и графической. При выборе литературы или графики используется выпадающий список, причем он будет содержать лишь те файлы, которые находятся на диске в соответствующей папке.

Вся информация (о сотрудниках, датах прохождения, вышеперечисленные файлы) находится на одном компьютере. Причин этому несколько:

- это позволяет облегчить задачу синхронизации базы данных (например, если один клиент поменял какие-то данные на одном компьютере, то их надо поменять и на других);
- такая организация обеспечивает большую безопасность, так как ограничить доступ к одному компьютеру всегда легче, чем к нескольким.

Организация сетевого взаимодействия. Так как данные хранятся на одном компьютере, необходимо реализовать сетевое взаимодействие между клиентом и сервером. Этапы взаимодействия:

1) клиент подключается к серверу;

2) сервер из базы данных формирует файл со списком отделов, должностей и сотрудников и отправляет его клиенту;

3) клиент получает файл, который является частью базы данных;

4) пользователь с компьютера-клиента выбирает отдел, должность и фамилию, причем данные берутся из полученного от сервера файла, и отправляет эти данные на сервер;

5) сервер формирует новый файл с тестовыми вопросами и картинками специально для каждого клиента и отправляет этот файл клиенту;

6) клиент распаковывает файл и ожидает разрешение для начала экзамена;

7) сервер дает разрешение и ожидает, когда пользователь приступит к экзамену;

8) клиент начинает экзамен, передав специальное сообщение серверу;

9) сервер производит наблюдение за ходом экзамена (включая информацию о правильных и неправильных ответах);

10) по окончании экзамена, если он успешен, сервер записывает дату прохождения экзамена в файл базы данных.

В режиме обучения наблюдение за его ходом не проводится, поэтому этапы взаимодействия в данном режиме ограничиваются пунктом номер 6, т.е. пользователь после передачи файла сразу отключается от сервера и приступает к обучению.

Для организации сетевого взаимодействия были выбраны компоненты *TClientSocket* и *TServerSocket*. Клиентский сокет включен в компонент *TClientSocket*. Слушающий сокет принимает запрос на соединение от клиентского сокета, и соединяет сервер с клиентом. Слушающий сокет содержится в компоненте *TServerSocket*. Серверный сокет обменивается данными с клиентом по уже установленному соединению.

Следует заметить, что для создания отдельных приложений клиента и сервера нет необходимости иметь несколько ЭВМ. На одной ЭВМ можно одновременно запускать и приложение-сервер и приложение-клиент. При этом в качестве адреса или имени ЭВМ, к которой будет производиться подключение, следует использовать имя *localhost* или IP-адрес *127.0.0.1*.

Сервер, основанный на сокетном протоколе, позволяет обслуживать сразу множество клиентов, причем ограничение на их количество можно устанавливать при программировании. Для каждого подключенного клиента сервер открывает отдельный сокет, по которому осуществляется обмен данными с клиентом.

Алгоритм управления сокетным сервером состоит из следующих этапов:

1) *установка свойств Port (порт подключения) и ServerType (тип подключения)* – для нормального взаимодействия сервера и клиента;

2) *открытие сокета и указанного порта* – после выполнения указанной операции выполняется автоматическое начало ожидания подсоединения клиентов (Listen);

3) *подключение клиента и обмен данными с ним*;

4) *отключение клиента*;

5) *закрытие сервера и сокета* – по команде администратора сервер завершает свою работу, закрывая все открытые сокетные каналы и прекращая ожидание подключений клиентов. Пункты 3 – 4 приведенного алгоритмы выполняются многократно, т.е. для каждого нового подключения клиента.

Заключение. Применение автоматизированной системы контроля знаний позволяет: устранить субъективные факторы, влияющие на результат контроля знаний; систематизировать данные о прохождении экзамена; устранить возможность фальсификации результатов путем надежной защиты информации с использованием криптографических алгоритмов шифрования.

ЛИТЕРАТУРА

1. Халин, Е.В. Информационная технология обеспечения безопасности производства / Е.В. Халин. – М.: ВИНИТИ, 1997. – 172 с.
2. Халин, Е.В. Компьютерные обучение и аттестация по безопасности производства / Е.В. Халин. – М.: НЕЛА-Информ, 2006. – 208 с.
3. Халин, Е.В. Безопасность производства: Технологии, способы, устройства / Е.В. Халин. – М.: ГНУ ВИЭСХ, 2006. – 372 с.
4. Спиридонов, А.В. Учебное программное обеспечение на основе тестовых сред / А.В. Спиридонов, Е.Р. Сухарев // Вестн. Полоц. гос. ун-та. Сер. Е. Педагогические науки. – 2006. – № 11. – С. 53 – 57.

Поступила 12.11.2008