Дипломная работа

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОЦЕНКИ УРОВНЯ ЗНАНИЙ СТУДЕНТОВ С ПРИМЕНЕНИЕМ ТЕХНОЛОГИИ «КЛИЕНТ-СЕРВЕР»

Москва 2010

# Введение

В настоящее время сфера образования стала одним из объектов внедрения вычислительной техники и информационных технологий.

ГОУ СПО «Тульский экономический колледж», является одним из среднеспециальных учебных заведений Тульской области.

В вычислительном центре ГОУ СПО «Тульский экономический колледж» имеются 117 компьютеров класса Pentium III и выше. Все они расположены в 7 учебных лабораториях. Также имеются необходимые периферийные устройства: струйный принтер Hewlett Packard Color Jet 100plus – формат А1, несколько лазерных принтеров; накопители на оптических и гибких дисках, DVD-RW. Все ПК оснащены мультимедийными устройствами (звуковые карты, приводы CD-ROM).

Компьютеры объединены в локальную сеть по средствам топологии «звезда» под управлением ОС Microsoft Windows 98/XP/Server2003.

Согласно требованиям современной педагогики, контроль успеваемости предполагается осуществлять в виде тестирования.

Тестирование можно проводить в виде опросных листов, что требует большой подготовки со стороны преподавателя. Наиболее трудоемким моментом является обработка результатов тестирования.

Темой данного дипломного проекта является «Разработка программного обеспечения для оценки уровня знаний студентов с применением технологии «Клиент-сервер».

# 1. Описание объекта автоматизации

В свободное от основной работы время, я занимаюсь преподавательской деятельностью на очном отделении ГОУ СПО «Тульский экономический колледж». Данный вид деятельности разрешен Законом о Государственной гражданской службе РФ.

Система управления колледжа традиционна для среднеспециального учебного заведения РФ.

Имеются следующие функциональные подсистемы:

* Директор;
* учебная часть: для организации учебного процесса, создания учебных планов, методических пособий, раздаточных материалов, ведение корреспонденции;
* отдел кадров: подготовка письменных документов таких как: приказы, справки, выписки, письма;
* вычислительный центр: проведение лабораторных и практических работ по получению первичных навыков работы на компьютере и профессиональных навыков по специальности, выполнение расчетов по практическим работам общеобразовательным и специальным дисциплинам, проверка знаний и умений учащихся (тестирование), проведение внеклассных мероприятий (занятий кружка, олимпиады).

Вычислительная техника в системе управления ГОУ СПО «Тульский экономический колледж» позволяет автоматизировать следующие функции управления:

* организация учебного процесса;
* ведение бухгалтерского учёта;
* ведение документации;
* ведение корреспонденции;
* учебный процесс.

Семь лабораторий вычислительного центра колледжа имеют по пятнадцать персональных компьютеров от Intel Celeron 533 Mhz до Intel Pentium IV 3000 Mhz каждый из них оснащен системой фильтрации от высокочастотных помех в цепи питания, накопителями FDD, устройствами ввода информации служат: клавиатуры и мыши. Также каждый из компьютеров подключен к сетевому принтеру в лаборатории.

Компьютеры объединены в сеть по средствам топологии «звезда» через коммутаторы (SWICH) фирмы D-Link.

Вычислительный центр обслуживается 4 лаборантами, которые закреплены за определёнными учебными лабораториями ВЦ. Они подчиняются зав. лабораторией и начальнику вычислительного центра.

# 2. Постановка задачи

## 

## **2.1 Сущность задачи**

Одним из важнейших структурных подразделений колледжа является вычислительный центр. Именно там проходят занятия учеников с использованием современных педагогических технологий, которые базируются на использовании ПК и программных средств.

Одним из этапов урока является проверка знаний и умений учащихся.

Согласно требованиям современной педагогики, контроль успеваемости предполагается осуществлять в виде тестирования.

Тестирование можно проводить в виде опросных листов, что требует большой подготовки со стороны преподавателя. Наиболее трудоемким моментом является обработка результатов тестирования.

Целью данного дипломного проекта является «Разработка программного обеспечения для оценки уровня знаний студентов с применением технологии «Клиент-сервер».

Актуальность поставленной задачи обусловлена облегчением труда преподавателя связанного с проведением тестирования и обработкой результатов данного тестирования.

Проведя исследование рынка программных продуктов по проведению сетевого тестирования, были выявлены ряд недостатков, в число которых входят:

– высокая стоимость отдельных программных продуктов

– невозможность работы с математическими формулами

– сложный для понимания пользовательский интерфейс

– большая загрузка ЛВС в момент проведения тестирования

Все вышеперечисленные недостатки послужили поводом для разработки собственного программного продукта.

Назначение программы – проведение централизованных итоговых занятий по разным дисциплинам в виде интерактивного тестирования.

Основными задачами дипломного проекта являются:

– разработка клиентской части системы тестирования

– разработка серверной части системы тестирования

– разработка системы отчетности по результатам тестирования

Функционирование автоматизированной системы тестирования, далее АСТ, начинается с создания для каждой дисциплины отдельной базы данных, содержащей вопросы теста, далее формируется «Тест пакет», содержащий всю информацию необходимую для осуществления тестирования с удаленной станции. В процессе тестирования появление вопросов происходит в случайном порядке. Возможности разработанного сетевого протокола позволяют вести двустороннее общение между преподавателем и студентом. Также программа позволяет осуществлять контроль, за ходом тестирования. Так в любой момент после начала тестирования преподаватель может посмотреть результаты тестирования каждого ученика (количество вопросов, на сколько из них были даны правильные ответы и сколько допущено ошибок), приостановить или прекратить тестирование, а также исключить отдельного ученика из процесса тестирования отключив его от сервера.

Выходной информацией АСТ является отчет успеваемости.

Периодичность применения автоматизированной системы зависит от плана преподавателя по проведению тестирования.

Для комфортной работы с программой необходим компьютер ниже перечисленной конфигурации:

* сетевая плата от 10 Мб/с и выше;
* процессор тактовой частотой не ниже 300 Мгц;
* оперативная память не менее 64 Мб;
* объем жесткого диска не менее 1 Гб;
* монитор 15 дюймов;
* разрешение монитора 1024x768 при 16 битной цветовой палитре;

# 3. Описание логической структуры

Система представляет собой совокупность двух подпрограмм осуществляющих проведения тестирования в рамках любой сети поддерживающей протокол TCP-IP.

Подпрограмма «Тест-Сервер» позволяет осуществлять управление над ходом тестирования студентов, она объединяет в себе возможности сетевого сервера, приложения и СУБД тестирования.

Подпрограмма «Тест-Клиент» позволяет осуществлять тестирование конкретного ученика, рабочая станция которого подключена к «Тест-Серверу» по заранее спроектированной схеме, способной динамически изменяться в ходе тестирования в соответствии с требованиями преподавателя.

Общая схема взаимодействия серверной и клиентской части системы приведена на рисунке 1.



Рис. 1. Общая схема взаимодействия серверной и клиентской части системы

Общая схема взаимодействия отдельный частей системы приведена на рисунке 2.

Действие пользователя

(тестируемого)

Обработка

действия пользователя

клиентской частью системы

Действие преподавателя. Контроль за ходом тестирования

Обработка

действия пользователя

СЕРВЕРНОЙ частью системы

Сетевой обмен данными от клиентской

части системы к ее серверной части

Сетевой обмен данными от серверной

части системы к ее клиентской части

Рис 2. Общая схема взаимодействия отдельный частей системы

## **3.1 Описание организации данных**

### 

### 3.1.1 Описание входной и выходной информации

Входной информацией является: Список вопросов, образующих билет и критерий оценки. Совокупность этих данных образуют промежуточное звено процесса тестирования – База Теста. Для обеспечения возможности сетевого тестирования необходима еще одна деталь – IP адрес станции, с которой будет осуществляться управление ходом тестирования, то есть станции, которая будет обозначена как ведущая и где будет развернут Тест-Сервер. Ввод информации необходимой для формирования Базы Теста осуществляется в главной форме подпрограммы Тест-Сервер на вкладке База вопросов.

База Теста представляет собой каталог, имеющий имя преподавателя по чьему предмету осуществляется тестирование. В данном каталоге располагаются файлы настроек для данной базы теста, а именно – файл QuestKey.ini – содержит номера правильных ответов для каждого вопроса; файл WorkSet.ini – содержит служебную информацию для данной базы теста, такую как: количество вопросов в тесте, ограничение времени для прохождения теста, формат изображений файлов вопросов теста. Основной каталог базы теста содержит N (зависящее от количества вопросов в тесте) дочерних каталогов имеющих системное имя в виде чисел от 1 до N, в которых содержатся изображения вопроса и вариантов ответов. Количество изображений может быть переменным (для осуществления случайного выбора вопроса теста), но должно быть не менее 1.



Рис. 3 Формирование базы теста

Каждое изображение вопросов и вариантов ответов должно иметь системное имя в виде сквозной нумерации от 1 до номера последнего вопроса. (например если всего 100 файлов вопросов, нумерация должна быть от 1 до 100). Все изображения должны иметь одинаковый формат, например, если некоторое количество файлов сделаны в формате JPEG Image file, то и все последующие должны быть в этом формате.

Структура каталогов базы теста приведена на рисунке 4.

Для клиентской части системы единственным файлом настроек является файл ip.dat, который содержит IP-адрес сервера тестирования

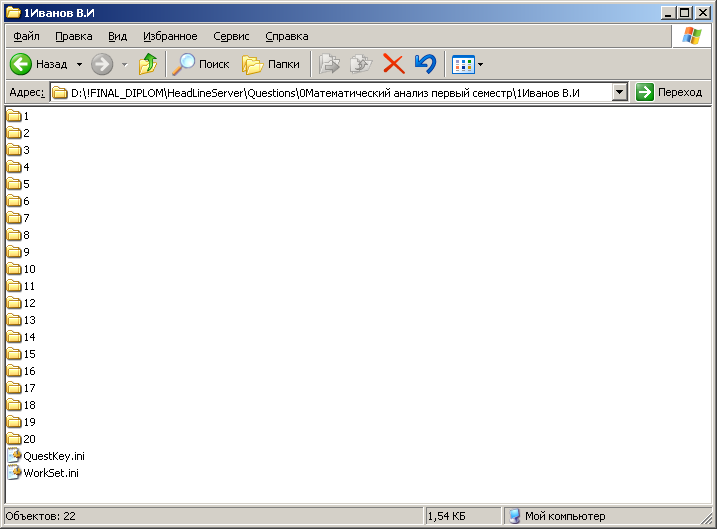


Рис. 4 Структура каталога базы теста

Основным элементом выходной информации является отчет успеваемости, имеющем следующие поля:

* Ф.И.О. студента;
* группа;
* общее количество вопросов;
* количество правильных ответов;
* количество не правильных ответов;
* оценка;
* время прохождения теста.

Предварительный просмотр отчета можно произвести из формы Отчет успеваемости, а также сохранить в файл и вывести на печать.

### 3.1.2 Система классификации и кодирования

Для эффективного обмена информацией между автоматизированными системами управления различных уровней требуется создание единого информационного обеспечения, включающего систему классификации и кодирования технико-экономической информации.

Система классификации и кодирования технико-экономической информации представляет собой комплекс взаимосвязанных общесоюзных классификаторов, а также комплекс нормативно-технических и методических материалов, характеризующих систему.

Основные работы по созданию системы классификации и кодирования технико-экономической информации выполняет Госстандарт.

Классификаторы по их применению делятся на категории:

– общегосударственные;

– отраслевые;

– классификаторы предприятия.

Общегосударственные – утверждаются ГОСТами и РОСТами и обязательны к применению во всех отраслях государства.

Отраслевые – разрабатываются для специфических видов информации, циркулирующей внутри отрасли и утверждаются министерствами и ведомствами. Обязательны к применению только в данной отрасли.

Классификаторы предприятия – аналогичны отраслевым, создаются и обязательны к применению на отдельных предприятиях.

В подсистеме «Учебная часть» используются следующие классификаторы:

1. Отраслевой классификатор для кодирования кода группы:

ХХХX – XХ

код названия специальности

код года поступления

код (флаг) некоммерческой группы

код текущего курса обучения

код (флаг) базы поступления

Пример: 0414-ФК – База 11 классов, четвертый курс, некоммерческая группа, последняя цифра года поступления, код специальности.

### 3.1.3 Защита и сохранность данных

Постоянное развитие компьютерной инфраструктуры, усложнение компьютерных систем, создание сетей, увеличение объема хранимой и передаваемой информации порождает ряд серьезных проблем, связанных с целостностью и сохранностью данных. Для обеспечения сохранности информации выделяют следующий комплекс мер:

* дублирование информации в виде резервных копий на том или ином носителе;
* защита от случайного удаления файлов;
* защита от несанкционированного доступа;
* защита от компьютерных вирусов;
* архивные копии;
* программный «уход» за жесткими дисками.

Для зашиты от несанкционированного доступа применяют различные методы защиты: процедурные, аппаратные, программные или комбинированные.

Процедурные методы обеспечивают доступ к данным только тем пользователям, которые имеют соответствующее разрешение.

Программные методы защиты очень разнообразны:

* использование ключевых меток на машинном носителе;
* использование серийных номеров программ;
* использование специального кода на инсталляционных дискетах.

Комбинированные методы защиты объединяют различные методы: процедурные и программные, аппаратные и программные и т.д.

Резервное копирование – это постоянное создание резервных копий рабочей информации. Существует много программ предназначенных для создания резервных копий. Из них наиболее часто используются Norton Backup и другие программы Backup из разных программных пакетов или Утилит, а также программа-архиватор WinRAR.

Так в программе «Оценка уровня знаний студентов с применением технологии «клиент-сервер» из пункта главного меню Файл \ Резервное сохранение можно создать резервную копию базы теста. Кроме этого дистрибутив программы, созданный с помощью специальной надстройки «InstallerXP», имеет несколько резервных копий на жестком диске рабочего места и на оптических дисках при главном компьютере ВЦ.

Чтобы информация, записанная на CD и RW дисках, сохранилась дольше, необходимо соблюдать определенные правила и требования: диски следует хранить в специальных коробках, упаковках или футлярах, предназначенных для этого, во избежание попадания пыли, физических повреждений, солнечных лучей.

Кроме создания резервных копий, для предотвращения потери информации на жестком или гибком диске, следует проводить обслуживание дисков: проводить полную проверку, то есть проверять на наличие физических ошибок, проверять структуру файлов и каталогов; если возникают ошибки, то исправлять их; производить дефрагментацию; удалять ненужную информацию. Обслуживание необходимо проводить регулярно. Для этого можно использовать следующие программные средства (утилиты) как Norton Disk Doctor (проверка жесткого диска), Norton Speed Disk (дефрагментация диска) из пакета Norton Utilities, Fix-It Utilities или Scandisk, который содержится в самой ОС Windows.

Часто информация теряется или повреждается в результате действий компьютерных вирусов.

Компьютерный вирус – это фрагмент программного кода, который размножается, копируя себя в тело других программ, при этом замедляется работа компьютера или полностью (либо частично) разрушается файловая система. Для того чтобы вирус не поразил компьютер необходимо: ограничить к нему доступ посторонних лиц, использующих различные носители информации; проверять на наличие вируса, если же он обнаруживается, то необходимо использовать антивирусные программы (Norton Antivirus, Antiviral Toolkit Pro, Dr. Web и др.). Антивирусные программы – это программы, написанные специально для выявления и уничтожения вирусов.

Еще один метод защиты программ от заражения вирусами – архивация данных.

Если наличие вируса очевидно (замедление работы жесткого диска, уменьшение скорости вычислительного процесса, появление несоответствующих данной задаче сообщений или картинок, внезапная потеря данных), то следует запустить антивирусную программу. Если она нашла вирус и излечила компьютер от него, то потеря информации будет минимальной или вообще ее может не быть. Если же не удалось вовремя излечить систему от вируса и после его действий почти вся информация была потеряна, то целесообразно произвести форматирование жесткого диска, чтобы полностью избавиться от файлов, зараженных вирусом. Затем загрузить систему с системной дискеты или компакт диска и перенести на жесткий диск информацию резервных копий.

Иногда требуется восстановить случайно уничтоженный файл. Для этого можно использовать программу UnErase из программного пакета Norton Utilities либо OnTrack Easy Recovery.

### 3.1.4 Организация и ведение информационной базы

Формирование базы теста осуществляется каждой дисциплины, ответственность за выполнение подготовки входной информация возлагается на преподавателя. По окончании подготовки оперативной информации необходимо ввести информацию в базу теста и сохранить готовую базу в автономный каталог с названием дисциплины и именем преподавателя.

С целью поддержания баз тестов в актуальном состоянии необходимо организовывать периодическое редактирование вопросов тестов в соответствии с требованиями учебного плана, ответственность за выполнение этой работы также возлагается на преподавателя.

Информация хранится в каталоге Questions, далее каталог «База Теста» и IP-адрес в файле IP.dat соответственно.

В каталоге База Теста содержатся вопросы теста варианты ответов на каждый вопрос, номер правильного ответа и порядковый номер вопроса. В форме осуществляющей ввод и редактирование информации Базы Теста предусмотрены активные кнопки и сопроводительные сообщения, которые позволяют улучшить эргономичность и избавиться от дополнительных элементов интерфейса. Для повышения производительности выбор верного ответа на вопрос осуществляется визуально, т.е. щелчком мыши на правильном ответе. В таблице IP-адрес содержится постоянная информация необходимая для осуществления сетевого подключения и функционирования всей сетевой подсистемы программы.

Массивы выходной информации после решения задачи сохраняются в файл. При необходимости сохранения информации, делается поименованная копия этой информации на любом доступном носителе. Срок хранения выходной информации определяется преподавателем.

Для связи выходной информации с другими задачами используется метод DDE – Dynamic Data Exchange, те обнуление, не требующихся для дальнейшего функционирования системы или принятия управленческого решения, данных и освобождение, таким образом, ресурсов происходит автоматически.

# 4. Описание программно-технических средств

## 

## **4.1 Программно-технические средства, необходимые для разработки программы**

Для разработки автоматизированной системы была выбрана платформа WINTEL под управлением операционной системы Windows XP SP2.

В качестве среды программирования для решения поставленной задачи была выбрана Borland Delphi 6.0 Enterprise.

Delphi – инструмент для создания приложений и систем, функционирующих на платформе Windows. В основе нее лежит объектно-ориентированный язык высокого уровня Object Pascal, разработанный профессором Высшего технического училища (г. Цюрих, Швейцария) Никлаусом Виртом.

Основными принципами ООП:

– инкапсуляцияпредставляет собой объединение данных и обрабатывающих их методов (подпрограмм) внутри класса. Это означает, что в классе инкапсулируются (объединяются и помещаются внутри класса) поля, свойства и методы. При этом класс приобретает определенную функциональность;

– наследованиезаключается в порождении новых объектов-потомков от существующих объектов родителей, при этом потомок берет от родителя все его поля, свойства и методы. В дальнейшем наследуемые поля, свойства и методы можно использовать в неизменном виде или переопределять (модифицировать). В новый объект добавляются новые элементы, определяющие его особенность и функциональность. Удалить какие-либо элементы родителя в потомке нельзя. В свою очередь, от нового объекта можно породить следующий объект, в результате образуется дерево объектов, или иерархия классов. В начале этого дерева находится базовый класс TObject, который реализует элементы, наиболее общие для всех объектов, например, действия по созданию и удалению объектов. Чем дальше тот или иной объект находится в дереве от базового класса, тем он более специфичен.

– полиморфизмзаключается в том, что методы различных объектов могут иметь одинаковые имена, но различное содержание. Это достигается переопределением родительского метода в классе-потомке. В результате родитель и потомок ведут себя по-разному. При этом обращение к одноименным методам различных объектов выполняется аналогично.

Следование стандартам индустрии и открытость к взаимодействию с любыми частными решениями гарантирует успех проектов, разрабатываемых с использованием Delphi.

Delphi устанавливает стандарт для сред разработки приложений Windows. Delphi обеспечивает набор возможностей специально ориентированных на многократное использование компонентов. Многие аспекты работы Delphi можно настраивать. Созданные полезные объекты – компоненты и шаблоны приложений и форм будут доступны для будущих разработок.

Delphi является первой системой RAD, в которой удачно соединились средства визуального проектирования и оптимизирующий компилятор, чего, к сожалению, нельзя сказать о других системах RAD. Delphi является единственным полноценным средством промышленной разработки систем клиент-сервер, на которой основывается и данная автоматизированная система контроля знаний.

В состав Delphi входит обширная библиотека компонентов, с помощью которой можно избежать ручного написания программ. С другой стороны, в любой момент можно прибегнуть к низкоуровневым ассемблерным процедурам. Можно создавать приложения в визуальном режиме. Работая в Delphi, можно с помощью нажатия одной клавиши создать исполняемый файл в формате EXE, однако, при необходимости, можно компилировать и файлы DLL, драйверов устройств, а также консольных приложений.

Существует множество достоинств, благодаря которым можно выделить Delphi из ряда других средств разработки:

* обширная библиотека классов;
* быстрый оптимизирующий компилятор, генерирующий машинный код;
* встроенный отладчик, равных которому нет;
* простой в освоении механизм доступа к базам данных;
* мощная и удобная в работе среда разработки.
* возможности Delphi, которые делают ее такой гибкой:
* прямой доступ к программному интерфейсу Windows;
* встроенный ассемблер и поддержка программирования в машинных кодах;
* возможность создания пользовательских компонентов VCL и ActiveX;
* поддержка формата DLL и других выполняемых файлов Windows;
* возможность многоуровневой разработки приложений;
* полная объективная ориентированность – в программах можно создавать объекты, берущие начало как от библиотечных классов, так и от созданных программистом.

Delphi предоставляет прямой доступ ко многим типам локальных и удаленных серверов баз данных. Также предоставляет множество различных типов для хранения целых, вещественных (с плавающей запятой), логических (boolean), символьных (char), строковых значений, а также указателей. Помимо этого имеются типы, определяемые пользователем: множества (sets), записи (records) и объектные переменные. Поскольку имеется столько разнообразных типов, понимание чужого программного кода может быть затруднено, если будут встречаться маловразумительные имена переменных.

Часто для обеспечения взаимодействия различных приложений или частей одного приложения организуется обмен данными. Для этого предоставляются следующие средства:

– использование буфера обмена;

– динамический обмен данными.

Буфер обмена представляет собой область оперативной памяти и специальных функций, которые используются для временного хранения файла. Буфер обмена является общим для всех программ, любое приложение может помещать в него информацию и считывать его оттуда. Буфер обмена способен хранить данные самых разных типов и содержит сведения об их формате. Буфер обмена обеспечивает простейший статический способ обмена данных между приложениями. Данные в общей области обмена обновляются и являются динамическими. Однако термин статический понимается в том смысле, что каждый раз, когда приложение или пользователь хочет получить новые данные из буфера обмена или поместить их туда, они должны выполнять для этого соответствующие операции.

Для выполнения операций обмена данными через буфер в Delphi предназначен специальный класс TClipBoard.

С помощью свойств и методов объекта Clipboard при работе с буфером обмена можно выполнить стандартные операции, например, очистить буфер или проанализировать тип хранимых данных. Для доступа к объекту буфера обмена в разделе Uses модуля, в котором выполняются операции с объектом буфера, указывается модуль Clipboard.

В Delphi создана поддержка технологии DDE (Dynamic Data Exchange – динамический обмен данными).

Динамический обмен данными (Dynamic Data Exchange – DDE) представляет собой технологию, которая связана с передачей данными между приложениями, работающими под управлением операционной системы Windows. С помощью технологии DDE два приложения могут динамически взаимодействовать и обмениваться текстовыми данными во время их выполнения. При этом изменения в одном приложении немедленно отражаются во втором приложении. Кроме того, с помощью технологии DDE можно из одного приложения управлять другим приложением, например, Microsoft Word или Excel.

При динамическом обмене два приложения соблюдают соглашение об обмене и устанавливают между собой непосредственную связь на время передачи данных. При этом программа, запрашивающая данные, становится клиентом, а программа, служащая источником данных, является сервером. В зависимости от направления передачи данных одно и то же приложение может одновременно быть и клиентом и сервером. Организация динамического обмена данными включает в себя два следующих этапа:

* установка связи между клиентом и сервером. Ее можно устанавливать при разработке и при выполнении приложения;
* передача текстовых данных, при этом возможны следующие действия:

– получение данных от сервера;

– передача данных на сервер;

– посылка серверу команд.

Delphi позволяет создавать оба типа приложений – сервера и клиента, при этом каждое из них создается отдельно. Для создания приложений, участвующих в динамическом обмене данными, существуют соответствующие компоненты. Для совместной отладки двух приложений можно сначала создавать сервер, а затем – клиент.

В качестве дополнительных средств, применяемых при реализации проекта можно отметить такие как Macromedia Flash MX – c помощью этого средства были созданы элементы анимации для некоторых процессов, растровый графический редактор Adobe Photoshop CS2 – его возможности помогли реализовать в проекте все неподвижные графические элементы.

Все данные приложения использовались в режиме TRIAL 30-дневной тестовой версии.

Аппаратная часть разработки проекта с учетом инструментальных средств предусматривает использование ПК следующей конфигурации:

– процессор тактовой частотой не ниже 700 Мгц;

– объем оперативной памяти не менее 128 Мб;

– диагональ монитора 15 и более дюймов;

– объем видеопамяти от 32 Мб;

– разрешение монитора 1024x768 при 16 битной палитре;

– объем жесткого диска не менее 2,1 Гб (1,5 Гб. – ОС Windows + 600 Mb – Borland Delphi 6.0).

## **4.2 Программно-технические средства, необходимые для эксплуатации программы**

Автоматизированной системы контроля знаний на основе архитектуры клиент-сервер работает в сетевом режиме. Для эксплуатации программы необходимы следующие программные средства:

Серверная часть:

* операционная система Windows 98 SE /Me/XP/2000/2003;
* пакет программ Microsoft Office XP (и последующие версии) для вывода отчетности.
* присутствие следующих компонентов операционной системы:
* сетевая плата либо контроллер удаленного доступа;
* протокол TCP/IP;

Клиентская часть:

* операционная система Windows 98 SE /Me/XP/2000/2003;
* присутствие следующих компонентов операционной системы:
* сетевая плата либо контроллер удаленного доступа;
* протокол TCP/IP;

Основные характеристики ОС Windows XP:

– многозадачность (одновременно может работать несколько приложений);

– работа с сетью ОС (на уровне ядра системы организован клиент / сервер сети);

– изоляция процессов (если во время работы какое-либо приложение совершило сбой, и в результате было закрыто аварийно, то это не сказывается на работе других приложений и процессов системы);

– поддержка огромного количества оборудования (в том числе, устаревшего и современного) всех известных производителей;

– широкие возможности настройки многих узлов системы (графических, интерфейсных, сетевых и т.д.);

– обширная справочная система по многим узлам операционной системы.

Использование Автоматизированной системы контроля знаний предусматривает следующие требования к аппаратным средствам:

* сетевая плата от 10 Мб/с;
* сетевая среда (физический уровень);
* процессор тактовой частотой не ниже 500 Мгц;
* оперативная память не менее 64 Мб;
* объем жесткого диска не менее 2 Гб;
* монитор 15 дюймов;
* разрешение монитора 1024x768 при 16 битной палитре;

## **4.3 Тестирование программы**

Для тестирования отдельных модулей-подпрограмм и автоматизированной системы в целом на ряду со стандартными интегрированными средствами тестирования и отладки, предоставляемые разработки Borland Delphi 6.0. (build 5.62) – Integrated Debugger, были применены и дополнительные средства, такие как Borland WinSight, Spy32 for Windows9x/NT, NuMega BoundsChecker, Registry Monitor Sysinternals Corp.

Для осуществления отладки при помощи Integrated Debugger необходимо активировать эту систему, с этой целью на странице Debugger Options пункта меню Tools среды Delphi был установлен флаг Integrated Debugging.

Реализация и тестирование отдельных модулей происходила в следующем порядке:

* разработка алгоритма решения задачи модуля в целом;
* руководствуясь разработанным алгоритмом, реализация отдельных подпрограмм и методов;
* тестирование отдельных подпрограмм и методов в автономном режиме, с проверкой входных и возвращаемых значений;
* компоновка подпрограмм в отдельный модуль;
* проверка синтаксиса модуля в целом;
* компиляция модуля;
* обнаружение и исправление ошибок в работе отдельных подпрограмм и при необходимости возврат к пункту 3;
* проведение функционального тестирования с целью выявления ошибок при работе с данными из области допустимых значений, граничными (находящиеся на границе области допустимых значений), выходящими за границу области допустимых значений.
* переход к разработке следующего модуля.

По окончании разработки и тестирования отдельных модулей проводилась их компоновка в систему модулей составляющих собственно программу в целом, после чего осуществлялось совместное тестирование модулей с целью выявления их взаимной несовместимости на отдельных этапах функционирования.

С этой целью активно использовались средство интегрированной отладки Integrated Debugger в состав которого входят такие функции как трассировка со входом в подпрограмму (Trace Into), пошаговое выполнение программы (Step Over), использование точек останова в коде (SourceBreakPoint), использование точек останова по адресу (AddressBreakPoint) просмотр значений идентификаторов при помощи WatchList и использование альтернативного, но более функционального средства DebugInspector.

По окончании разработки, тестирования, отладки и конкатенации модулей в единую систему, с образованием исполняемого модуля проводилось структурное тестирование функций программного комплекса в целом. С этой целью производилась как последовательная, так и перекрестная активация всех функциональных подсистем комплекса. Для внешнего контроля за корректностью работы автоматизированной системы, с точки зрения операционной системы Windows, были использованы дополнительные средства тестирования.

Borland WinSight – использовался для визуализации иерархической системы окон проекта и исследовании потока системных сообщений в адрес элементов управления проекта.

Еще одно инструментальное средство, которое использовалось при структурном тестировании проекта – Spy32 for Windows9x/NT. Программа Spy32 позволила протестировать функционирование отдельных элементов интерфейса путем обращения к их обработчикам на уровне системных сообщений.

Корректность совместной работы проекта с менеджером памяти Windows позволил осуществить программный комплекс NuMega BoundsChecker. Правильность обращений к реестру были проконтролированы при помощи Registry Monitor от Sysinternals Corp.

## **4.4 Описание программы**

Автоматизированная система для оценки уровня знаний студентов с применением технологии «Клиент-сервер» предназначена для проведения централизованных итоговых занятий по разным дисциплинам в виде интерактивного тестирования.

Данный проект поддерживает совместимость с пакетом Microsoft Office в применении единого формата данных и обеспечивает передачу данных в стандартные средства MS Office, такие как MS Word и MS Excel.

Автоматизированная система представляет собой совокупность двух программ HL Server и HL Client.

Программа HL Server – предназначена для координации процесса тестирования, формирования и редактирования базы теста, генерации информации необходимой для осуществления тестирования.

HLClient – программа, предназначена для осуществления двусторонней связи с ведущим компьютером для диалога с преподавателем, передачи информации о ходе тестирования каждого ученика и служебных данных обеспечивающих корректную работу сетевой подсистемы проекта.

Установка АСТ начинается с запуска хранителя дистрибутива. Далее необходимо следовать инструкциям по установке.

В процессе установки все файлы, необходимые для функционирования АСТ, помещаются в системную директорию («C:\Programm Files\HLTest»), имеющую при успешной установке пакета, следующее структурное содержание:

* файл HLServer.exe – главный исполняемый модуль;
* каталог Groups – содержит текстовые файлы имеющие системное имя студенческой группы, содержимое данного файла – список фамилий студентов данной группы;
* каталог Questions – содержин каталоги с наименованием дисциплин, каждый из которых содержит Базу Теста в виде каталога, имеющего имя преподавателя по данному предмету.

Запуск программы HLServer можно осуществить по выбору при помощи созданного, в процессе установки, ярлыка HLTest.lnk на рабочем столе, либо выбором в меню Пуск – Программы – HLTest – HLServer.lnk.

При запуске происходит выделение необходимых для ее функционирования ресурсов и инициализация переменных окружения, таких как параметры интерфейса и рабочая директория, локальный IP адрес.

В зависимости от целей запуска программы HLServer можно начать работать над созданием или ведением базы теста, формированием пакета теста или же начать тестирование.

Перед началом тестирования необходимо проверить работоспособность сети в целом, позаботиться о распространении (при помощи стандартных сетевых средств операционной системы) и активации пакета теста путем запуска программы HLClient.exe на всех компьютерах.

При завершение работы HLServer происходит инструктирование всех подключенных рабочих станций (если таковые имеются) о необходимости в освобождении занимаемых ими ресурсов.

При шаговом переходе по записям производится контроль над выходом за границы массива записей, при котором выводится соответствующее сообщение. Для удобства перехода к необходимым билетам, организован визуальный навигатор, позволяющий быстро перейти к нужному билету. Использование навигатора позволяет обойти алгоритм контроля над выходом за границы массива записей.

При формировании нового билета происходит контроль над вводимыми значениями, так при не заполнении одного из полей билета или отсутствии выбора верного ответа, выдается сообщение о необходимости дополнить веденную информацию.

При выборе подпункта «Закрыть» на экран выводится диалог, позволяющий выбрать между закрытием базы теста и отменой действия. При положительном ответе происходит закрытие текущей базы теста, освобождение занимаемых ресурсов и вывод соответствующего сообщения.

При выборе подпункта «Удалить» на экран выводится диалог, позволяющий выбрать между удалением базы теста и отменой действия. При положительном ответе происходит удаление и оповещение о результате выполнения операции.

При подключении новых клиентов к серверу происходит сетевой запрос информации о клиенте, ответ на который включает следующие поля:

* Ф.И.О. студента;
* группа обучения;
* статус станции (готов к тестированию, проходит тестирование, окончил тестирование);
* количество верных ответов;
* количество ошибок;
* общее число пройденных билетов;
* IP – адрес станции.

Существует возможность в любой момент отключить станцию от сервера, для этого необходимо выбрать станцию подлежащую отключению и щелкнуть на кнопке – «Отключить» в главной форме.

Предусмотрена возможность остановки тестирования по времени. При вводе времени, по истечению которого, тестирование должно прекратиться, происходит контроль за корректностью ввода формата времени, так если в поле содержащем количество секунд будет введено число большее чем 60, будет выведено сообщение об ошибке при вводе времени.

Предварительный просмотр отчета успеваемости можно просмотреть щелкнув на кнопке «Отчет успеваемости» в главной форме при этом откроется форма содержащая отчет успеваемости, информацию из которой, можно направить в MS Word для дальнейшей распечатки.

Функционирования автоматизированной системы на клиентской стороне начинается с активации пакета теста путем запуска программы HLClient.exe входящей в состав пакета.

После запуске программы HLClient в центре экрана появляется форма, в которой тестируемый студент должен выбрать код группы, в которой он проходит обучение и свою фамилию, пока соответствующие поля не будут заполнены продолжение работы программы невозможно. По окончании выбора необходимо нажать на кнопку «Начать тестирование», что приведет к появлению на экране формы, при появлении которой программа HLClient производит инициализацию и загрузку сетевых параметров системы и на основе полученных данных осуществляет попытку соединения с сервером теста.

При соединении с сервером происходит двунаправленная пересылка данных, необходимых для дальнейшего функционирования программ HLClient и HLServer как единой автоматизированной системы.

После прохождения теста осуществляется подсчет числа верных ответов, ошибок и сравнение полученных значений с критерием оценки. Вывод оценки осуществляется в виде сообщения на главной форме. По прошествии 6 секунд после появления сообщения с оценкой, на экран выводится сообщение «Ждите окончания тестирования».

Далее клиентская сторона АСТ вновь переходит в режим ожидания сигнала-окончания тестирования, при получении которого происходит закрытие программы HLClient.

# 5. Мероприятия по охране труда, технике безопасности и противопожарной защите

В целях обеспечения охраны труда и техники безопасности в ГОУ СПО «Тульский экономический колледж» предусмотрены следующие меры:

* все видеотерминалы соответствуют требованиям европейского стандарта TCO'99;
* в соответствии с европейскими требованиями по организации рабочего места с использованием вычислительной техники и во избежание повреждения внутренних и внешних устройств вычислительной от статического электричества произведено их заземление;
* используются сетевые фильтры и источники бесперебойного питания;
* используются как внутренние, так и внешние кондиционеры воздуха.

На подоконниках располагаются растения, у которых под действием солнечной энергии (лучей света) протекает процесс фотосинтеза, в результате чего листья растений поглощают углекислый газ (СО2) и выделяют кислород (О2), тем самым фильтруя воздух в лаборатории.

В соответствии с ГОСТ 12.1.005–88 «Общие санитарно-гигиенические требования к воздуху рабочей зоны», в рабочих помещениях постоянно поддерживается постоянная температура воздуха 20–24 градусов по шкале Цельсия, относительная влажность 40–60% и скорости движения воздуха не более 0,1 м/с.

Перепад температуры по высоте рабочей зоны допускается до 3С0, по горизонтали до 5С0.

Интенсивность теплового облучения работающих от нагретых поверхностей оборудования, осветительных приборов не должна превышает 35 Вт/кв. м при облучении 50% поверхности тела, 70 Вт/кв. м – при величине облучаемой поверхности от 25% до 50% и 100 Вт/кв. м при облучении не более 25% поверхности тела.

Уровень электромагнитного излучения установлен ГОСТом 12.1.006–84 ССБТ. «Электромагнитные поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению контроля».

Уровни инфракрасногои ультрафиолетового излучения установлены ГОСТом 27.016–86. «Дисплей; Трубки электронно-лучевые приемные».

Уровень шума, создаваемый вентиляционной системой не превышает допустимого уровня в 70 Дба, уровень шума от принтера HP 690C соответствует ISO 9296 и составляет 50 Дба, что не превышает 55 Дба установленных в ГОСТ 12.1.003–83. «Шум. Общие требования безопасности».

Каждый день проводится влажная уборка, что обеспечивает соответствие с требованиями СН 512–78 «Инструкция по проектированию зданий и помещений ЭВМ» запыленность воздуха в рабочем помещении, и не превышая предельно допустимуюконцентрацию(ПДК), равную 2 мг/куб. м, и находится в пределах нормы, специально для этой работы имеется технический персонал в количестве одного человека.

Все компьютерное оборудование, используемое в ГОУ СПО «Тульский экономический колледж» питается от сети переменного тока напряжением 220В, частотой 50Гц.

Техника безопасности при работе с электрооборудованием направлена, прежде всего, на предотвращение случаев поражения электрическим током. Перед первым включением компьютера следует проверить, соответствует ли напряжение в сети тому, на которое рассчитан компьютер (многие компьютеры могут работать при нескольких значениях входного напряжения, например при 220 В или 110 В). При необходимости надо установить переключатель напряжения на компьютере в правильное положение.

Хотя встроенные в компьютеры блоки питания, преобразующие напряжение электросети в низковольтное напряжение (±12 В и ±5 В), достаточно устойчивы и работают даже при понижении или повышении напряжения на 10–15%, однако далеко не всегда обеспечивают безопасность компьютеров и их устойчивую работу. С этой целью в ГОУ СПО «Тульский экономический колледж» на питание на компьютеры подаётся через средства защиты от недостатков электропитания подключенных по следующей схеме: стабилизатор Штиль 500 -> сетевой фильтр Pilot 1200F -> бесперебойного питания APC Back UPS CS 500.

Наибольшую опасность представляет двухполюсное прикосновение человека к токоведущим частям, когда он оказывается под полным рабочим напряжением сети.

Допустимые значения напряжения прикосновения и тока, протекающего через тело человека, регламентированы ГОСТом 12.038–82 и оцениваются по трем критериям электробезопасности:

1. Ощутимый ток. Не вызывает нарушения деятельности организма и допускается для длительного протекания. Сила тока для переменного тока 0,3 МА. Сила тока для постоянного тока 1 МА.

2. Отпускающий ток. Его действие на человека допустимо, если длительность протекания не превышает 30 секунд. Сила для переменного тока 6 МА. Сила для постоянного тока 15 МА.

3. Фибриляционный ток. Допускается воздействие не более 1 секунды. Сила для переменного тока 50 МА. Сила для постоянного тока 200 МА.

Технические и организационные меры защиты обеспечивают недоступность токопроводящих частей и невозможность случайного прикосновения к ним, устраняют опасность поражения при замыкании на корпус или на землю.

Техника безопасности электрооборудования включает в себя следующие виды защиты от поражения током:

– ограждение токопроводящих частей;

– блокировку;

– предупредительные надписи;

– двойную изоляцию корпусов;

* изоляцию токопроводящих частей;
* защитное разделение питания;
* защитное заземление металлических частей корпусов.

Освещенность рабочего места должна удовлетворять требованиям СНиП 11–4–79 «Строительные нормы. Нормы проектирования. Естественное и искусственное освещение». Освещение при работе с ЭВМ должно быть смешанным: естественное (боковое или прямое) и искусственное (общее). Искусственное освещение должно осуществляться с помощью люминесцентных ламп. При естественном освещении должны применяться средства солнцезащиты (светорассеивающие шторы или жалюзи). Для исключения бликов отражения на экране применяются антибликовые покрытия, специальные фильтры и другие средства предусмотренные заводом изготовителем оргтехники в целях обеспечения соответствия с европейским стандартом TCO'99.

Для освещения кабинета используются естественные и искусственные формы, то есть в дневное, не пасмурное, время суток в качестве естественной формы освещения выступает Солнце, а в качестве искусственной формы освещения используются люминесцентные лампы. Для поступления солнечных лучей в кабинетах имеются три окна со светорассеивающими шторами.

В соответствии с требованиями СНиП 11–4–79 уровень освещенности на рабочем месте при искусственном освещении должен быть не менее 300 лк в кабинетах колледжа этот показатель составляет в среднем 323 лк.

Элементы рабочего места – рабочее кресло и стол должны соответствовать требованиям ГОСТа 12.2.032–78 и ГОСТа 21889–76.

Площадь помещений для работников ВЦ следует предусматривать величиной не менее 6.0 м^2 на человека. В соответствии с ВСНиП 4559–88 рабочий стол должен регулироваться по высоте в пределах 680–760 мм. Оптимальные размеры рабочей поверхности столешницы 1600x900 мм. Под столешницей рабочего стола должно быть свободное пространство для ног с размерами по высоте не менее 600 мм, по ширине 500 мм, по глубине 650 мм. На поверхности рабочего стола для документов необходимо предусмотреть размещение специальной подставки, расстояние которой от глаз должно быть аналогичным расстоянию от глаз до клавиатуры, что позволяет снизить зрительное утомление.

Рабочий стул (кресло) должен обеспечивать удобную посадку при работе и обладало следующими функциональными возможностями:

* Устойчивое основание, например, пять ножек на роликах. Конструкция ножек должна соответствовать поверхности пола, независимо от того, гладкий он или с ковровым покрытием.
* Возможность регулировки кресла по высоте и углу наклона. Высота кресла должна легко регулироваться в пределах от 40 до 52 см (считается расстояние от пола до сиденья). Если рост выше или ниже среднего, может потребоваться кресло с большим диапазоном регулировки. Необходимо отрегулировать кресло таким образом, чтобы рабочая поверхность или полка для клавиатуры были на уровне локтя, ступни ног полностью стояли на полу, а колени согнуты и расположены немного ниже бедер. В идеальном случае сиденье кресла должно наклоняться как вперед (минимум на 5 градусов), так и назад (минимум на 10 градусов). Если кресло имеет регулируемое сиденье, необходимо наклонить его немного вперед. Это перенесет нагрузку с позвоночника на бедра и ступни, уменьшив нагрузку на спину.
* Изогнутый край сиденья. Передняя часть сиденья должна быть изогнута и иметь мягкие края.
* Регулируемая по высоте и наклону спинка. Очень важно, чтобы спинка плотно прилегала к нижней части спины (пояснице).
* Легко вращающееся основание, которое позволяет свободно поворачиваться в обе стороны.
* Полностью регулируемые и мягкие подлокотники. Подлокотники не должны препятствовать регулировке кресла и мешать вплотную придвигаться к рабочей поверхности.

На рабочем месте необходимо предусматривать подставку для ног.

В соответствии с эргономическими требованиями, установленными ГОСТом 21829–76 ВДТ должен отвечать следующим требованиям:

* яркость свечения экрана не менее 100 кд/м2;
* минимальный размер светящейся точки – не более 0.4 мм;
* контрастность изображения знака – не менее 0.8;
* экран должен иметь антибликовое покрытие.

В настоящее время установлены жесткие стандарты безопасности на современные мониторы, что понижает риск ухудшения самочувствия при работе с ними. Клавиатура не должна быть жестко связана с монитором.

Большой опасностью для любого предприятия являются пожары, наносящие немалые убытки. Главными причинами возникновения пожаров на производстве являются:

* короткое замыкание;
* повышенная температура окружающих предметов;
* неисправное электрооборудование или неправильная его эксплуатация.

Пожарная профилактика – это комплекс организационных и технических мероприятий, направленных на обеспечение безопасности людей, на предотвращение пожара, ограничение его распространения, а также на создание условий для успешного тушения пожара.

Во избежание пожара или взрыва, необходимо строго выполнять противопожарные меры безопасности.

Устройство и эксплуатация оборудования, зданий и сооружений должны соответствовать требованиям противопожарной безопасности.

Для прекращения процесса горения используются следующие основные меры:

– Охлаждение горящих веществ путем нанесения на их поверхность теплоемких огнетушащих средств (воды, пены и т.д.) или перемешивания слоев горящей жидкости.

– Разбавление концентрации горючих паров, пыли и газов путем введения в зону горения инертных газов (азот, углекислый газ).

– Изоляция горящих веществ от зоны горения нанесением на их поверхность изолирующих огнегасительных средств (пены, песка, кашмы).

– Химическое торможение реакции горения путем орошения поверхности горящих материалов или объемного разбавления горючей пыле – газо- и паровоздушной системы флегматизирующими веществами и составами.

Противопожарные меры ГОУ СПО «Тульский экономический колледж» заключаются в следующем:

* в коридорах висят планы эвакуации в случае пожара;
* имеется пожарный кран в коридорах на каждом этаже и огнетушители типа ОХВП-10 в каждом кабинете.

# 6. Экономическая часть

## 

## **6.1 Расчет себестоимости программного продукта**

В экономической части дипломного проекта рассчитывается себестоимость программного продукта.

Исходные данные для расчёта себестоимости программного продукта приведены в таблице 1.

Таблица 1. Исходные данные

|  |  |
| --- | --- |
| Наименование | Значение |
| Количество форм входной информации  Из них:  переменной  нормативно-справочная  банка данных | 3  2  0  1 |
| Количество разновидностей форм выходной информации | 1 |
| Степень новизны задачи | Б |
| Сложность алгоритма | 1 |
| Вид используемой информации | ПИ |
| Сложность контроля:  входной информации  **выходной информации** | 12  22 |
| Язык программирования | Borland Delphi 6.0 |
| Вид обработки | Режим реального времени (РВ) |

Алгоритм расчёта себестоимости программного продукта:

анализ исходных данных;

расчёт трудоёмкости разработки программного продукта;

расчёт стоимости машинного часа;

расчёт стоимости программного продукта;

Расчёт трудоёмкости разработки программного продукта по стадиям представлен в Таблице 2.

Таблица 2. Трудоёмкость разработки по стадиям

|  |  |  |  |
| --- | --- | --- | --- |
| Стадия разработки проекта | Затраты времени | Поправочный коэффициент | Затраты времени с учётом поправочного коэффициента |
| Значение | Значение |
| 1 | 2 | 3 | 4 |
| 1. Разработка технического задания | | | |
| 1.1. Затраты времени разработчика постановки задачи | 15 | 0,65 | 9,5 |
| 1.2. Затраты времени разработчика программного обеспечения | 15 | 0,35 | 5,25 |
| 2. Разработка эскизного проекта | | | |
| 2.1. Затраты времени разработчика постановки задачи | 34 | 0,7 | 23,8 |
| 2.2. Затраты времени разработчика программного обеспечения | 34 | 0,3 | 10,2 |
| 3. Разработка технического проекта | | | |
| 3.1. Затраты времени разработчика постановки задачи | 9 | 1,827 | 16,443 |
| 3.2. Затраты времени разработчика программного обеспечения | 5 | 1,827 | 9,135 |
| 4. Разработка рабочего проекта | | | |
| 4.1. Затраты времени разработчика постановки задачи | 3 | 3,6936 | 11,0808 |
| 4.2. Затраты времени разработчика программного обеспечения | 27 | 3,6936 | 99,7272 |
| 5. Внедрение | | | |
| 5.1. Затраты времени разработчика постановки задачи | 5 | 1,39 | 6,95 |
| 5.2. Затраты времени разработчика программного обеспечения | 5 | 1,39 | 6,95 |

Время работы ЭВМ при отладке и внедрении программы складывается из затрат времени разработчика программного обеспечения на технический проект, рабочий проект и внедрение.

Таким образом, затраты времени на отладку и внедрение составляют 49 человеко-дней или 392 часов.

### 6.1.1 Расчёт стоимости одного машинного часа.

Стоимость одного машинного часа определяется по формуле

#### , где (7)



*Эксп* – эксплуатационные годовые затраты (в рублях);

*Тф* – количество часов, отработанных всеми машинами в год (час).

Эксплуатационные годовые затраты включают в себя:

1. Годовая амортизация оборудования (Аоб), формула 8;
2. Годовые затраты на ремонт оборудования (Роб), формула 9;
3. Расходы на электроэнергию (Зэл), формула 10;
4. Прочие расходы (Зпр), формула 14.

1. Годовая амортизация оборудования определяется:

#### , где (8)



*Косн* – коэффициент амортизации основного оборудования (в процентах);

*Сосн* – стоимость основного оборудования (в рублях);

*Квсп* – коэффициент амортизации вспомогательного оборудования (в процентах);

*Свсп* – стоимость вспомогательного оборудования (в рублях).

В данном случае стоимость одного принтера пропорционально распределена между двумя компьютерами.

руб.



2. Годовые затраты на текущий ремонт составляют 5% от общей стоимости используемого оборудования.

, где (9)



*Собщ* – общая стоимость оборудования (в рублях).

руб.



3. Затраты на электроэнергию складываются из расходов на освещение *Вос* (формула 10) и расходов на производственное потребление электроэнергии *Вэ* (формула 11).

*Зэл=Вос+Вэ,* где (10)

*Вос* – расходы на освещение (в рублях);

*Вэ* – расходы на производственное потребление электроэнергии (в рублях).

, где (11)



*S* – площадь помещения (в квадратных метрах);

*Кэ* – усреднённый расход энергии, для освещения одного квадратного метра площади помещения в год (кВт на квадратный метр);

*Стар* – тариф (в рублях).

руб.



, где (12)



*Нуст* – мощность одного компьютера (кВт);

*Н* – количество компьютеров (штук);

*К* – коэффициент учитывающий потери в сети;

*Стар* – тариф (в рублях);

*Ф* – годовой фонд времени работы оборудования рассчитывается по формуле:

#### , где (13)



*Нг* – число дней в году;

*Нвых* – число выходных дней в году;

*Нпр* – число праздничных дней в году;

*Ксм* – коэффициент сменности;

*Фдн* – продолжительность рабочего дня;

*Кзаг* – коэффициент загрузки оборудования;

*Крем* – коэффициент, учитывающий потери времени на ремонт оборудования.

часа.



Тогда расходы на производственное потребление электроэнергии (по формуле 12) равны руб.



Затраты на электроэнергию (по формуле 10) равны руб.



4. Прочие расходы составляют 5% от суммы расходов по предыдущим пунктам.

#### , где (14)



#### *Аоб* – сумма годовой амортизации (в рублях);

#### *Робщ* – годовые затраты на ремонт (в рублях);

#### *Э* – расходы на электроэнергию (в рублях).



#### руб.



#### Тогда эксплуатационные годовые расходы составляют:

#### , где (15)



#### *Аоб* – сумма годовой амортизации (в рублях);

#### *Робщ* – годовые затраты на ремонт (в рублях);

#### *Э* – расходы на электроэнергию (в рублях);

#### *Зпр* – прочие расходы (в рублях).

руб.



Количество часов, отработанных всеми машинами в год равно:

#### , где (16)



*Н* – количество компьютеров (в штуках);

*Ф* – годовой фонд времени работы оборудования (в часах).

#### часов



#### Тогда стоимость одного машинного часа (по формуле 7) равна:

#### руб.



### 

### 6.1.2 Расчёт стоимости программного продукта.

Стоимость программного продукта определяется по формуле:

, где (17)



*Тдн* – затраты времени на разработку (чел.-дней);

*Змес* – среднемесячная зарплата (в рублях);

*Ндн* – количество рабочих дней в месяце (дни);

*Тмаш* – затраты времени на отладку и внедрение (в часах);

*См.ч.* – стоимость одного машинного часа (в рублях).

##### руб.



# Заключение

В данном дипломном проекте представлена «Автоматизированная система контроля знаний на основе архитектуры клиент-сервер», реализованная в среде программирования Borland Delphi 6.0.

Дополнительные средства разработки и возможности среды программирования позволили осуществить формирование и ведение базы теста, вывод необходимых форм и отчета успеваемости, создать удобный пользовательский интерфейс включающий:

* стандартная строка меню;
* кнопки – для активизации функций системы;
* сопроводительные сообщения.

Для повышения надежности хранения информации предусмотрены программные средства защиты информации:

* резервное сохранение базы теста;

Наличие встроенной контекстной помощи позволяет упростить использование программы.

Дипломный проект был выполнен в заданный срок.

# Приложение 1

# Листинг кода серверной части программы

program HLServer;

uses

Forms,

BaseUnit in 'BaseUnit.pas' {MainForm},

QBaseWork in 'QBaseWork.pas',

UBaseWork in 'UBaseWork.pas';

{$R \*.res}

begin

Application. Initialize;

Application. CreateForm (TServerForm, ServerForm);

Application. Run;

end.

unit BaseUnit;

interface

uses

QBaseWork, UBaseWork, Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ScktComp, Grids, StdCtrls, ExtCtrls, Menus, CommCtrl, ComCtrls,

IniFiles, WinSock, ComObj, OleServer, Word97, ShellCtrls, Buttons, Word2000;

const

NM\_Register1 = 6; // прием списка групп

NM\_Register2 = 7; // запрос на список студентов

NM\_RegisterGetWorks = 66; // запрос / ответ 'список предметов'

NM\_RegisterGetTeachers = 77; // запрос / ответ 'список преподователей'

NM\_RegisterOK = 8; // клиент зарегистрирован

NM\_Service = 31; // прием сервисной информации

NM\_TestEvent = 55; // событие по ходу тестирования

NM\_FileOperation = 10; // сетевая операция с файлами

NM\_EndOfTest = 33; // окончание тестирования

NM\_KickFromServer = 44; // отключение от сервера администратором

NM\_OutOfTime = 50; // отключение по истечении времени

NM\_DataError = 54; // проблема с БД

NM\_Wait = 61;

type

PCustomWinSocket=TCustomWinSocket;

Questions=record // Структура вопроса

Passed:boolean; // пройден (да/нет)

Style:byte; // стиль вопроса {radio, check, memo}

UserAnswer: word; // ответ пользователя

TrueAnswer: word; // верный ответ

end;

PathID=record

WorkID:byte;

TeacherID:byte;

end;

Peoples=record // структура 'Пользователь'

SocketHandle: Integer; // дескриптор соединения

Ip:string[15]; //IP адрес

Num:byte; // номер клиента

Registered:boolean; // прошел регистрацию (да/нет)

TestingAbortedByTime:boolean;

Group:string[8]; // группа

Name:string[20]; // имя

Teacher:string[40]; // преподаватель

WorkName:string[40]; // наим. дисциплины

WorkPath:string[255]; // рабочая директория пользователя

UserWorkPathID: PathID; // идентификаторы дисциплины и преподавателя

ImageType:string[3]; // тип файла вопросов {зарезервировано}

QuestCount:byte; // количество вопросов

OpenQuest:byte; // Ссылка на билет из массива Questions

 // для дальнейшего

TimeLater:TTime; // потрачено времени

SumTime:TTime; // общий бюджет бремени

PassedCount:byte; // пройдено вопросов

True\_:byte; // верных ответов

False\_:byte; // неверных ответов

Mark:byte; // оценка

PassTest:boolean; // тест пройден (да/нет)

Questions:array [1..255] of Questions; // массив пройденных вопросов

end;

type

TServerForm = class(TForm)

ServerSocket1: TServerSocket;

PageControl1: TPageControl;

TabSheet1: TTabSheet;

ComboBox1: TComboBox;

ListBox1: TListBox;

Label2: TLabel;

Label3: TLabel;

Timer1: TTimer;

Label4: TLabel;

Label5: TLabel;

TabSheet4: TTabSheet;

ConnectionCount: TLabel;

Timer2: TTimer;

TabSheet8: TTabSheet;

Panel3: TPanel;

Button3: TButton;

Button4: TButton;

Image1: TImage;

RadioGroup1: TRadioGroup;

ShellTreeView1: TShellTreeView;

ShellListView1: TShellListView;

ComboBox2: TComboBox;

Bevel8: TBevel;

Label1: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Label16: TLabel;

Label10: TLabel;

Label17: TLabel;

Label18: TLabel;

Bevel1: TBevel;

Bevel4: TBevel;

Bevel5: TBevel;

Bevel6: TBevel;

Bevel7: TBevel;

Bevel9: TBevel;

Bevel13: TBevel;

Bevel10: TBevel;

Bevel11: TBevel;

Bevel12: TBevel;

Bevel14: TBevel;

Bevel15: TBevel;

Bevel16: TBevel;

Bevel17: TBevel;

Bevel18: TBevel;

Bevel19: TBevel;

Bevel20: TBevel;

WordDocument1: TWordDocument;

SpeedButton1: TSpeedButton;

PageControl2: TPageControl;

TabSheet3: TTabSheet;

TabSheet5: TTabSheet;

StringGrid1: HLringGrid;

StringGrid2: HLringGrid;

TabSheet6: TTabSheet;

Memo1: TMemo;

Button7: TButton;

Button8: TButton;

SaveDialog1: TSaveDialog;

Panel2: TPanel;

Label29: TLabel;

Label30: TLabel;

Label31: TLabel;

Label32: TLabel;

TabSheet7: TTabSheet;

ReportGrid: HLringGrid;

Button1: TButton;

procedure ServerSocket1ClientConnect (Sender: TObject;

Socket: TCustomWinSocket);

procedure FormCreate (Sender: TObject);

procedure FormDestroy (Sender: TObject);

procedure ServerSocket1ClientRead (Sender: TObject;

Socket: TCustomWinSocket);

procedure ComboBox1Change (Sender: TObject);

procedure Timer1Timer (Sender: TObject);

procedure ServerSocket1ClientDisconnect (Sender: TObject;

Socket: TCustomWinSocket);

procedure Timer2Timer (Sender: TObject);

procedure StringGrid1DblClick (Sender: TObject);

procedure Button3Click (Sender: TObject);

procedure ShellListView1Change (Sender: TObject; Item: TListItem;

Change: TItemChange);

procedure ShellListView1DblClick (Sender: TObject);

procedure Image1Click (Sender: TObject);

procedure ShellTreeView1Enter (Sender: TObject);

procedure ServerSocket1ClientError (Sender: TObject;

Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;

var ErrorCode: Integer);

procedure Button1Click (Sender: TObject);

procedure SpeedButton1Click (Sender: TObject);

procedure StringGrid1SelectCell (Sender: TObject; ACol, ARow: Integer;

var CanSelect: Boolean);

procedure Button7Click (Sender: TObject);

procedure Button8Click (Sender: TObject);

private

function DecodeNumToSocketNum (StationNum: byte): byte;

procedure SendQuestion (ForStation: byte; TheFile: String; QuesHLyle:byte; TrueAnswer: Word);

procedure TestEvent (StationNum: byte; Socket\_:PCustomWinSocket);

procedure SendFileMessage (var Message: TMessage); message WM\_USER;

procedure LogMessage (var Message: TMessage); message WM\_USER+2;

procedure FillReportTable;

procedure CreateReport;

procedure TableClear (Table:HLringGrid);

procedure ReFillTable;

procedure CriticalClientDisconnect (Ip, Name, Group, WorkName,

TeacherName: String; TrueAnsw, FalseAnsw: byte; TimeLater: TTime);

procedure TimeRefresh;

procedure ProblemWithData (From\_:PCustomWinSocket; TxtMessage: string);

procedure AddLogMessage (Message\_: string);

procedure DisconnectComboBoxUpdate;

procedure TimeOUTTesting (StationNum: byte);

 // function DecodeSocketToClientNum (Socket\_: THandle): byte;

end;

var

ServerForm: TServerForm;

FOptions:TIniFile;

NetworkErrors:word;

RootPath:string;

DataSetForReport:array [0..44] of Peoples;

CurrenHLation:byte;

GroupList: String;

RegisteredClients:byte;

PassedTestCount:byte;

ConnectedSumm:byte;

 // TimeForPassTest:TTime;

SelectedRow:integer;

CurrentQuestFile:string;

CurrentQuestionNum:integer;

DoAction:boolean;

QUESTIONBASE:TQuestDB;

USERSBASE:TUsersDB;

SecCounter:byte;

Processing:boolean;

implementation

{$R \*.dfm}

procedure TServerForm. SendQuestion (ForStation:byte; TheFile: String; QuesHLyle: Byte; TrueAnswer: Word); // Отправка вопроса

var FileStream:TMemoryStream; // Файловый поток

Command:byte; // Команда

procedure LoadFileForSend (const FileName: string); // Локальная процедура подготовки

var Stream: HLream; // файлового потока

Count: Int64; // размер файла данных

MakePointer:DWORD; // искусственный указатель

CurrSize: Int64; // размер файлового потока

FNameLen:byte; // длина имени файла (для корректного распознавания на стороне клиента)

begin

Stream:= TFileStream. Create (FileName, fmOpenRead or fmShareDenyWrite); // создаем поток

try

Count:= Stream. Size;

Stream. Position:=0;

 // далее переносим информацию в поток

FileStream. WriteBuffer (Count, SizeOf(Int64)); // размер файла данных

FNameLen:=Length(FileName);

FileStream. WriteBuffer (FNameLen, 1); // длина имени файла

FileStream. WriteBuffer (Pointer(FileName)^, FNameLen); // имя файла

FileStream. Position:=0;

CurrSize:=FileStream. Size;

FileStream. SetSize (Count+CurrSize); // расширяем поток (в смысле размера)

MakePointer:=DWORD (FileStream. Memory)+CurrSize;

if Count<>0 then Stream. ReadBuffer (Pointer(MakePointer)^, Count); // переписываем данные из потока в поток

 // с использованием указателя на память

finally

Stream. Free; // освобождаем промежуточный поток

end;

end;

begin

try

Command:=NM\_FileOperation;

FileStream:=TMemoryStream. Create;

FileStream. WriteBuffer (Command, 1);

FileStream. WriteBuffer (TrueAnswer, 2);

FileStream. WriteBuffer (QuesHLyle, 1);

LoadFileForSend(TheFile);

FileStream. Position:=0;

ServerSocket1. Socket. Connections[ForStation].SendStream(FileStream); // отправка потока

except

FileStream. Free;

end

end;

 // очищать неверный дисконнект

procedure TServerForm. SendFileMessage (var Message: TMessage); // внутреннее событие отправка файла

var

DataStream:TMemoryStream;

Data:byte;

StationNum:byte;

PSock:TCustomWinSocket;

begin

StationNum:=Message.WParam;

if DataSetForReport[StationNum].PassedCount=0 then

begin

DataStream:=TMemoryStream. Create; // создаем поток

Data:=NM\_Service; // код команды

DataStream. WriteBuffer (Data, 1);

Data:=DataSetForReport[StationNum].QuestCount; // количество вопросов

DataStream. WriteBuffer (Data, 1);

DataStream. WriteBuffer (DataSetForReport[StationNum].SumTime, SizeOf (DataSetForReport[StationNum].SumTime)); // время на тестирование

DataStream. Position:=0;

ServerSocket1. Socket. Connections [DecodeNumToSocketNum(StationNum)].SendStream(DataStream);

 // отправка потока

sleep(1); // задержка 1ms

end;

PSock:=ServerSocket1. Socket. Connections [DecodeNumToSocketNum(StationNum)];

TestEvent (StationNum,@PSock); // генерация события связанного с тестированием

end;

function TServerForm. DecodeNumToSocketNum (StationNum:byte):byte; // поиск индекса станции в динамическом

var TryConnectedStation:byte; // массиве Connections по известному

begin // по номеру

Result:=0;

if DataSetForReport[StationNum].SocketHandle<>0 then

for TryConnectedStation:=ServerSocket1. Socket. ActiveConnections-1 downto 0 do // перебираем все соединения

begin // поиск ведется по дескриптору соединения

if ServerSocket1. Socket. Connections[TryConnectedStation].SocketHandle=DataSetForReport[StationNum].SocketHandle then

begin

Result:=TryConnectedStation; // если найдена соответствующая станция,

break; // выходим предварительно

end;

end;

end;

procedure TServerForm. ServerSocket1ClientError (Sender: TObject; // ошибка соединения

Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;

var ErrorCode: Integer);

begin

ErrorCode:=0;

DoAction:=true;

Inc(NetworkErrors);

Socket. Close;

end;

Procedure TServerForm. AddLogMessage (Message\_:string);

begin

SendMessage (Handle, WM\_User+2, DWord (PChar(Message\_)), 0);

end;

procedure TServerForm. ServerSocket1ClientConnect (Sender: TObject; // соединение

Socket: TCustomWinSocket);

var ConnectionsScan:byte;

ConnectedClientNum:byte;

Buff:string;

Command:byte;

ConnectOK:boolean;

procedure KickFromServer;

begin

Command:=NM\_KickFromServer;

Socket. SendBuf (Command, 1);

end;

begin

AddLogMessage (Socket. RemoteAddress+' Has client connection, check Socket…');

ConnectOK:=false;

if ServerSocket1. Socket. ActiveConnections<=45 then // если сервер не заполнен

begin

for ConnectionsScan:=0 to 44 do // ищем пустую ячейку (т. к. кто-то мог отсоединится)

begin

if (DataSetForReport[ConnectionsScan].SocketHandle=0) and (not (DataSetForReport[ConnectionsScan].PassTest)) then // если нашли сохраняем ее номер и идем дальше

begin

ConnectedClientNum:=ConnectionsScan;

DataSetForReport[ConnectionsScan].SocketHandle:=Socket. SocketHandle; // Заполняем ячейку буфера соединений

DataSetForReport[ConnectionsScan].Num:=ConnectedClientNum;

Buff:=Char (NM\_Register1)+Char(ConnectionsScan)+GroupList+'>'; // список групп и персональный номер

Socket. SendBuf (Pointer(Buff)^, Length(Buff)); // отправка буфера

CurrenHLation:=ConnectedClientNum;

ConnectOK:=true;

AddLogMessage (Socket. RemoteAddress+' Client accepted');

break;

end;

end;

end else AddLogMessage (Socket. RemoteAddress+' Server is Full');

if not ConnectOK then

begin

AddLogMessage (Socket. RemoteAddress+' Client not accepted');

KickFromServer;

end;

Inc(ConnectedSumm); // увеличиваем счетчик соединений

end;

procedure TServerForm. CriticalClientDisconnect (Ip:string; Name, Group, WorkName, TeacherName: String; TrueAnsw, FalseAnsw:byte; TimeLater:TTime);

var i:byte;

begin

if Ip<>'' then

for i:=1 to StringGrid2. RowCount-1 do

begin

if StringGrid2. Cells [0, i]='' then

begin

StringGrid2. RowCount:=i+2;

StringGrid2. Cells [0, i]:=Ip;

StringGrid2. Cells [1, i]:=Name+' '+Group;

StringGrid2. Cells [2, i]:=WorkName;

StringGrid2. Cells [3, i]:=TeacherName;

StringGrid2. Cells [4, i]:=IntToStr (TrueAnsw+FalseAnsw);

StringGrid2. Cells [5, i]:=IntToStr(TrueAnsw);

StringGrid2. Cells [6, i]:=IntToStr(FalseAnsw);

StringGrid2. Cells [7, i]:=TimeToStr(TimeLater);

break;

end;

end;

end;

procedure TServerForm. ServerSocket1ClientDisconnect (Sender: TObject;

Socket: TCustomWinSocket);

var ScanConnections:byte;

DisconnectedClientNum:integer;

begin

for ScanConnections:=44 downto 0 do // перебираем все возможные подключения

begin

if DataSetForReport[ScanConnections].SocketHandle=Socket. SocketHandle then // ищем отключившуюся станцию

begin

DisconnectedClientNum:=ScanConnections;

if not DataSetForReport[DisconnectedClientNum].PassTest then // Если станция отключилась до окончания тестирования

 // то исключить ее из отчета

begin

AddLogMessage (Socket. RemoteAddress+' Client critical disconnect');

CriticalClientDisconnect (

DataSetForReport[DisconnectedClientNum].Ip,

DataSetForReport[DisconnectedClientNum].Name,

DataSetForReport[DisconnectedClientNum].Group,

DataSetForReport[DisconnectedClientNum].WorkName,

DataSetForReport[DisconnectedClientNum].Teacher,

DataSetForReport[DisconnectedClientNum].True\_,

DataSetForReport[DisconnectedClientNum].False\_,

DataSetForReport[DisconnectedClientNum].TimeLater

);

DataSetForReport[DisconnectedClientNum].Name:='';

if DataSetForReport[ScanConnections].Registered then

begin

Dec(RegisteredClients);

DataSetForReport[ScanConnections].Registered:=false;

DisconnectComboBoxUpdate;

end;

ZeroMemory (Addr(DataSetForReport[DisconnectedClientNum].Questions), 254);

break;

end;

AddLogMessage (Socket. RemoteAddress+' Client pass test and disconnect');

DataSetForReport[ScanConnections].PassedCount:=0;

DataSetForReport[ScanConnections].SocketHandle:=0; // обнуляем соответствующую ячейку

DataSetForReport[ScanConnections].Num:=0;

ConnectionCount.caption:=inttostr(ConnectedSumm);

DoAction:=true;

break;

end;

end;

Dec(ConnectedSumm);

if ConnectedSumm=0 then AddLogMessage (' Server is empty');

end;

procedure TServerForm. ServerSocket1ClientRead (Sender: TObject;

Socket: TCustomWinSocket);

type TDataBuffer=array of byte;

var

Command:byte; // собственно команда

SendLen:integer; // Длина всего принятого потока

DataBuffer:TDataBuffer;

ClientNum:byte;

FieldNum:byte;

NameBuf:string;

SendBuff:string;

BuffLen:integer;

OpenedBuilet:byte;

UserAnswer: Word;

Wait:byte;

Procedure SetMark;

begin

if DataSetForReport[ClientNum].Questions[OpenedBuilet].TrueAnswer=UserAnswer then

begin

inc (DataSetForReport[ClientNum].True\_);

inc (DataSetForReport[ClientNum].Mark);

end

else inc (DataSetForReport[ClientNum].False\_);

end;

begin

Wait:=NM\_Wait;

if not Processing then

begin

SendLen:=Socket. ReceiveLength;

SetLength (DataBuffer, SendLen);

ZeroMemory (DataBuffer, SendLen);

Socket. ReceiveBuf (Pointer(DataBuffer)^, SendLen);

Command:=DataBuffer[0];

ClientNum:=DataBuffer[1];

case Command of

NM\_Register2:

begin

USERSBASE. SetActiveGroup (DataBuffer[2]);

SendBuff:=Char (NM\_Register2)+USERSBASE. GetUsersStringList;

BuffLen:=Length(SendBuff);

Socket. SendBuf (Pointer(SendBuff)^, BuffLen);

end;

NM\_RegisterGetWorks:

begin

SendBuff:=Char (NM\_RegisterGetWorks);

SendBuff:=SendBuff+QUESTIONBASE. GetWorksStringList;

BuffLen:=Length(SendBuff);

Socket. SendBuf (Pointer(SendBuff)^, BuffLen);

end;

NM\_RegisterGetTeachers:

begin

FieldNum:=DataBuffer[2]; // номер элемента списка

NameBuf:='';

QUESTIONBASE. TransactionUser:=Socket. RemoteAddress+' name unknown';

if QUESTIONBASE. SetActiveWork(FieldNum) then

begin

NameBuf:=QUESTIONBASE. ActivWorkName;

SendBuff:=Char (NM\_RegisterGetTeachers)+SendBuff+QUESTIONBASE. GetTeachersStringList;

BuffLen:=Length(SendBuff);

Socket. SendBuf (Pointer(SendBuff)^, BuffLen);

end else ProblemWithData (@Socket, 'Error with Database');

end;

NM\_RegisterOK:

begin

{

0 – команда

1 – № клиента

2 – Группа

3 – Ф.И.О.

4 – WorkName

5 – Teacher

}

 // 1 {определение группы}

{РЕГИСТРАЦИЯ}

DataSetForReport[ClientNum].Group:=USERSBASE. GetGroupByIndex (DataBuffer[2]);

if (USERSBASE. SetActiveGroup (DataBuffer[2])) and (USERSBASE. SetActiveUser (DataBuffer[3])) then

begin

DataSetForReport[ClientNum].Ip:=Socket. RemoteAddress;

DataSetForReport[ClientNum].Name:=USERSBASE. ActiveUserName;

QUESTIONBASE. TransactionUser:=Socket. RemoteAddress+' '+DataSetForReport[ClientNum].Name+' '+DataSetForReport[ClientNum].Group;

 // 3 {определение дисциплины}

if (QUESTIONBASE. SetActiveWork (DataBuffer[4])) then

if (QUESTIONBASE. SetActiveTeacher (DataBuffer[5])) then

begin

DataSetForReport[ClientNum].QuestCount:=QUESTIONBASE. QuestionsCount;

DataSetForReport[ClientNum].WorkName:=QUESTIONBASE. GetWorkByIndex (DataBuffer[4]);

DataSetForReport[ClientNum].UserWorkPathID. WorkID:=DataBuffer[4];

 // 4 {определение имени руководителя}

DataSetForReport[ClientNum].Teacher:=QUESTIONBASE. GetTeacherByIndex (DataBuffer[5]);

DataSetForReport[ClientNum].UserWorkPathID. TeacherID:=DataBuffer[5];

DataSetForReport[ClientNum].SumTime:=StrToTime (QUESTIONBASE. WorkTimeLimit);

AddLogMessage (Socket. RemoteAddress+' '+DataSetForReport[ClientNum].Name+' '+DataSetForReport[ClientNum].Group+' Client passed registration');

DataSetForReport[ClientNum].Ip:=Socket. RemoteAddress;

DataSetForReport[ClientNum].True\_:=0;

DataSetForReport[ClientNum].False\_:=0;

DataSetForReport[ClientNum].Mark:=0;

DataSetForReport[ClientNum].TestingAbortedByTime:=false;

DataSetForReport[ClientNum].TimeLater:=StrToTime ('0:00:00');

DataSetForReport[ClientNum].PassTest:=false;

DataSetForReport[ClientNum].WorkPath:=RootPath+'Questions\'+DataSetForReport[ClientNum].WorkName+'\'+DataSetForReport[ClientNum].Teacher;

DataSetForReport[ClientNum].PassedCount:=0;

DataSetForReport[ClientNum].ImageType:=QUESTIONBASE. ImgFileType;

DataSetForReport[ClientNum].Registered:=true;

DisconnectComboBoxUpdate;

CurrenHLation:=ClientNum;

Inc(RegisteredClients); // зарегистрировано клиентов

PostMessage (Handle, WM\_USER, ClientNum, 0);

DoAction:=true;

end else

begin

ProblemWithData (@Socket, 'Error with Database');

AddLogMessage (Socket. RemoteAddress+' Problem with registration, client application shutdown');

end;

end else

begin

ProblemWithData (@Socket, 'Error with Database');

AddLogMessage (Socket. RemoteAddress+' Problem with registration, client application shutdown');

end;

end;

NM\_TestEvent:

begin

UserAnswer:=DataBuffer[2];

OpenedBuilet:=DataSetForReport[ClientNum].OpenQuest;

DataSetForReport[ClientNum].Questions[OpenedBuilet].Passed:=true;

Inc (DataSetForReport[ClientNum].PassedCount);

if DataSetForReport[ClientNum].QuestCount=DataSetForReport[ClientNum].PassedCount then

begin // если пройдены все билеты то заканчиваем тестирование

DataSetForReport[ClientNum].PassTest:=true;

SetMark;

inc(PassedTestCount);

SendBuff:=Char (NM\_EndOfTest)+Char (DataSetForReport[ClientNum].Mark);

ZeroMemory (Addr(DataSetForReport[ClientNum].Questions), 254);

BuffLen:=Length(SendBuff);

Socket. SendBuf (Pointer(SendBuff)^, BuffLen);

end else SetMark;

PostMessage (Handle, WM\_USER, ClientNum, 0);

DoAction:=true;

end;

end;

end else

begin

Socket. SendBuf (Wait, 1);

beep;

end;

end;

procedure TServerForm. TimeOUTTesting (StationNum:byte);

var SendBuff:string;

BuffLen:integer;

begin

DataSetForReport[StationNum].TestingAbortedByTime:=true;

DataSetForReport[StationNum].PassTest:=true;

inc(PassedTestCount);

SendBuff:=Char (NM\_EndOfTest)+Char (DataSetForReport[StationNum].Mark);

ZeroMemory (Addr(DataSetForReport[StationNum].Questions), 254);

BuffLen:=Length(SendBuff);

ServerSocket1. Socket. Connections [DecodeNumToSocketNum(StationNum)].SendBuf (Pointer(SendBuff)^, BuffLen);

end;

procedure TServerForm. TableClear (Table:HLringGrid);

var i:word;

begin

for i:=1 to Table. RowCount do Table. Rows[i].Clear;

end;

procedure TServerForm. ReFillTable;

var i, ii:byte;

begin

DoAction:=false;

TableClear(StringGrid1);

i:=1;

if RegisteredClients>=StringGrid1. RowCount then StringGrid1. RowCount:=StringGrid1. RowCount+1;

for ii:=0 to 44 do

begin

if (DataSetForReport[ii].Registered) and (not DataSetForReport[ii].PassTest) then

begin

StringGrid1. Cells [0, i]:=DataSetForReport[ii].Ip;

StringGrid1. Cells [1, i]:=DataSetForReport[ii].Name;

StringGrid1. Cells [2, i]:=DataSetForReport[ii].Group;

StringGrid1. Cells [3, i]:=IntToStr (DataSetForReport[ii].True\_+DataSetForReport[ii].False\_);

StringGrid1. Cells [4, i]:=IntToStr (DataSetForReport[ii].True\_);

StringGrid1. Cells [5, i]:=IntToStr (DataSetForReport[ii].False\_);

StringGrid1. Cells [7, i]:=TimeToStr (DataSetForReport[ii].SumTime-DataSetForReport[ii].TimeLater);

StringGrid1. Cells [6, i]:=TimeToStr (DataSetForReport[ii].TimeLater);

StringGrid1. Cells [8, i]:='в процессе';

inc(i);

end;

end;

Label10. Caption:=IntToStr(PassedTestCount);

Label17. Caption:=IntToStr(NetworkErrors);

ConnectionCount. Caption:=inttostr(ConnectedSumm);

Label18. Caption:=IntToStr (RegisteredClients-PassedTestCount);

Label16. Caption:=IntToStr(RegisteredClients);

end;

procedure TServerForm. TimeRefresh;

var i, ii:byte;

begin

i:=1;

for ii:=0 to 44 do

begin

if (DataSetForReport[ii].Registered) and (not DataSetForReport[ii].PassTest) and (not DataSetForReport[ii].TestingAbortedByTime) then

begin

StringGrid1. Cells [6, i]:=TimeToStr (DataSetForReport[ii].TimeLater);

StringGrid1. Cells [7, i]:=TimeToStr (DataSetForReport[ii].SumTime-DataSetForReport[ii].TimeLater);

inc(i);

end;

end;

end;

procedure TServerForm. FormCreate (Sender: TObject);

var NewSearch:TSearchRec;

begin

QUESTIONBASE:=TQuestDB. Create(Handle);

USERSBASE:=TUsersDB. Create(Handle);

RootPath:=ExtractFilePath (Application. ExeName);

ShellTreeView1. Root:=RootPath+'Questions\';

StringGrid1. Cells [0,0]:='IP адрес';

StringGrid1. Cells [1,0]:='ФИО';

StringGrid1. Cells [2,0]:='Группа';

StringGrid1. Cells [3,0]:='Пройдено билетов';

StringGrid1. Cells [4,0]:='Верных';

StringGrid1. Cells [5,0]:='Неверных';

StringGrid1. Cells [6,0]:='Время тестирования';

StringGrid1. Cells [7,0]:='Осталось времени';

StringGrid1. Cells [8,0]:='Статус';

ReportGrid. Cells [0,0]:='ФИО';

ReportGrid. Cells [1,0]:='Группа';

ReportGrid. Cells [2,0]:='Дисциплина';

ReportGrid. Cells [3,0]:='Преподаватель';

ReportGrid. Cells [4,0]:='Верных';

ReportGrid. Cells [5,0]:='Неверных';

ReportGrid. Cells [6,0]:='Время';

ReportGrid. Cells [7,0]:='Оценка';

StringGrid2. Cells [0,0]:='IP адрес';

StringGrid2. Cells [1,0]:='ФИО';

StringGrid2. Cells [2,0]:='Дисциплина';

StringGrid2. Cells [3,0]:='Преподаватель';

StringGrid2. Cells [4,0]:='Пройдено';

StringGrid2. Cells [5,0]:='Верных';

StringGrid2. Cells [6,0]:='Неверных';

StringGrid2. Cells [7,0]:='Время';

GroupList:=USERSBASE. GetGroupsStringList;

FindFirst ('Groups\\*.txt', faAnyfile, NewSearch);

repeat

Delete (NewSearch. Name, Length (NewSearch. Name) – 3,4);

ComboBox1. Items. Add (ExtractFileName(NewSearch. Name));

until FindNext(NewSearch)<>0;

if GroupList='' then ShowMessage ('Нет списков групп сервер незапущен') else ServerSocket1. Active:=true;

FindClose(NewSearch);

end;

procedure TServerForm. FormDestroy (Sender: TObject);

begin

ServerSocket1. Close;

ServerSocket1. Active:=false;

QUESTIONBASE. Destroy;

USERSBASE. Destroy;

end;

 ////////////////

procedure TServerForm. Timer1Timer (Sender: TObject);

var StationNum:byte;

begin

if (ConnectedSumm >0) or (StringGrid1. Cells [0,1]<>'') then

begin

if SecCounter>5 then

begin

DoAction:=true;

SecCounter:=0;

end else inc(SecCounter);

if RegisteredClients>0 then

for StationNum:=44 downto 0 do

if (DataSetForReport[StationNum].Registered) and (not DataSetForReport[StationNum].PassTest) and (not DataSetForReport[StationNum].TestingAbortedByTime) then

begin

DataSetForReport[StationNum].TimeLater:=DataSetForReport[StationNum].TimeLater+StrToTime ('0:00:01');

if DataSetForReport[StationNum].TimeLater>=DataSetForReport[StationNum].SumTime then TimeOUTTesting(StationNum);

end;

if DoAction then

begin

ReFillTable;

FillReportTable;

end else TimeRefresh;

end else ConnectionCount.caption:=inttostr(ConnectedSumm);

end;

procedure TServerForm. ProblemWithData (From\_:PCustomWinSocket; TxtMessage:string);

var SendBuf:string;

BuffLen:byte;

begin

SendBuf:=Char (NM\_DataError);

SendBuf:=SendBuf+Char (Length(TxtMessage))+TxtMessage;

BuffLen:=Length(SendBuf);

From\_.SendBuf (Pointer(SendBuf)^, BuffLen);

end;

procedure TServerForm. TestEvent (StationNum:byte; Socket\_:PCustomWinSocket);

var CurrenHLation: Peoples;

WorkPath:string;

TmpStr: String;

SumCount: Byte;

RNDQuestNum: Word;

TrueAnsw: Word;

begin

CurrenHLation:=DataSetForReport[StationNum];

WorkPath:=DataSetForReport[StationNum].WorkPath;

SumCount:=DataSetForReport[StationNum].QuestCount;

randomize;

if DataSetForReport[StationNum].PassedCount<SumCount then

begin

QUESTIONBASE. TransactionUser:=DataSetForReport[StationNum].Ip+' '+DataSetForReport[StationNum].Name+' '+DataSetForReport[StationNum].Group;

repeat

RNDQuestNum:=random(SumCount)+1; // Случайный номер вопроса

until not DataSetForReport[StationNum].Questions[RNDQuestNum].Passed;

if QUESTIONBASE. SetActiveWork (DataSetForReport[StationNum].UserWorkPathID. WorkID) then

if QUESTIONBASE. SetActiveTeacher (DataSetForReport[StationNum].UserWorkPathID. TeacherID) then

begin

TmpStr:=QUESTIONBASE. GetRandomFileBuilet(RNDQuestNum);

if TmpStr<>'' then // Случайный билет

 // Найти верный ответ и послать по сети

begin

TrueAnsw:=QUESTIONBASE. GetTrueAnswerForBuilet(TmpStr);

 // |–Вычисляем номер сокета клиента

 // \/

SendQuestion (DecodeNumToSocketNum(StationNum), TmpStr, 0, TrueAnsw);

DataSetForReport[StationNum].OpenQuest:=RNDQuestNum;

DataSetForReport[StationNum].Questions[RNDQuestNum].Style:=0;

DataSetForReport[StationNum].Questions[RNDQuestNum].Passed:=False;

DataSetForReport[StationNum].Questions[RNDQuestNum].TrueAnswer:=TrueAnsw;

DataSetForReport[StationNum].Questions[RNDQuestNum].UserAnswer:=0;

end else ProblemWithData (Socket\_, 'Error with Database');

end else ProblemWithData (Socket\_, 'Error with Database');

end;

end;

 //////////////////////

 /////////////////////

 ////////////////////

procedure TServerForm. ComboBox1Change (Sender: TObject);

var fNames:textfile;

NameBuf:string;

NameCounter:byte;

begin

ListBox1. Clear;

AssignFile (fNames, 'Groups\'+ComboBox1. Items [ComboBox1. ItemIndex]+'.txt');

{$i-}

Reset(fNames);

NameCounter:=0;

While not Eof(fNames) do

begin

Readln (fNames, NameBuf);

ListBox1. Items. Add (IntToStr(NameCounter)+' '+NameBuf);

inc(NameCounter);

end;

Label5. Caption:=IntToStr(NameCounter);

CloseFile(fNames);

{$i+}

end;

procedure TServerForm. Timer2Timer (Sender: TObject);

begin

Panel2. Visible:=false;

Timer2. Enabled:=false;

end;

procedure TServerForm. StringGrid1DblClick (Sender: TObject);

var MPoint:TPoint;

begin

if StringGrid1. Cells [0, SelectedRow]<>'' then

begin

GetCursorPos(MPoint);

MPoint:=ScreenToClient(MPoint);

Label31. Caption:=DataSetForReport [SelectedRow-1].WorkName;

Label32. Caption:=DataSetForReport [SelectedRow-1].Teacher;

panel2. Top:=MPoint.Y;

panel2. Left:=MPoint.X;

panel2. Visible:=true;

timer2. Enabled:=True;

end;

end;

procedure TServerForm. Button3Click (Sender: TObject);

var ExtNameLen:byte;

NumName:string;

NumN: Word;

StrCQFile:string;

TrueAsw:byte;

begin

if not Panel3.visible then

begin

ExtNameLen:=Length (ExtractFileExt(CurrentQuestFile));

NumName:=ExtractFileName(CurrentQuestFile);

Delete (NumName, Length(NumName) – ExtNameLen+1, ExtNameLen);

try

CurrentQuestionNum:=StrToInt(NumName);

TrueAsw:=QUESTIONBASE. GetTrueAnswerForBuilet(CurrentQuestFile);

RadioGroup1. ItemIndex:=TrueAsw-1;

RadioGroup1. Show;

except

ShowMessage ('Это не файл билета');

exit;

end;

Image1. Picture. Bitmap. LoadFromFile(CurrentQuestFile);

Panel3.visible:=true;

Button3. Caption:='Закрыть';

end else

begin

Panel3.visible:=false;

RadioGroup1. Visible:=False;

Button3. Caption:='Просмотреть билет';

RadioGroup1. Hide;

end;

end;

procedure TServerForm. ShellListView1Change (Sender: TObject;

Item: TListItem; Change: TItemChange);

begin

Button3.enabled:=false;

if ShellListView1. ItemIndex>=0 then

begin

CurrentQuestFile:=ShellTreeView1. Path+'\'+PChar (ShellListView1. SelectedFolder. DisplayName);

if (AnsiUpperCase (ExtractFileExt(CurrentQuestFile))=AnsiUpperCase ('.bmp')) or (AnsiUpperCase(ExtractFileExt(CurrentQuestFile))=AnsiUpperCase ('.jpg')) then Button3.enabled:=true;

end;

end;

procedure TServerForm. ShellListView1DblClick (Sender: TObject);

begin

Button3.enabled:=false;

if ShellListView1. ItemIndex>=0 then

begin

CurrentQuestFile:=ShellTreeView1. Path+'\'+PChar (ShellListView1. SelectedFolder. DisplayName);

if AnsiUpperCase (ExtractFileExt(CurrentQuestFile))=AnsiUpperCase ('.bmp') then

begin

Button3.enabled:=true;

Button3. Click;

end;

end;

end;

procedure TServerForm. Image1Click (Sender: TObject);

begin

Button3. Click;

end;

procedure TServerForm. ShellTreeView1Enter (Sender: TObject);

begin

Button3. Enabled:=false;

end;

procedure TServerForm. FillReportTable;

var i, ii:byte;

begin

i:=1; // начинаем со второй строки

TableClear(ReportGrid);

if PassedTestCount>0 then

begin

for ii:=0 to 44 do

begin

if (DataSetForReport[ii].PassTest) then

begin

ReportGrid. Cells [0, i]:=DataSetForReport[ii].Name;

ReportGrid. Cells [1, i]:=DataSetForReport[ii].Group;

ReportGrid. Cells [2, i]:=DataSetForReport[ii].WorkName;

ReportGrid. Cells [3, i]:=DataSetForReport[ii].Teacher;

ReportGrid. Cells [4, i]:=IntToStr (DataSetForReport[ii].True\_);

ReportGrid. Cells [5, i]:=IntToStr (DataSetForReport[ii].False\_);

ReportGrid. Cells [6, i]:=TimeToStr (DataSetForReport[ii].TimeLater);

ReportGrid. Cells [7, i]:=IntToStr (DataSetForReport[ii].Mark);

inc(i);

end;

ReportGrid. RowCount:=i+2;

end;

end else ShowMessage ('Нет прошедших тестирование');

end;

procedure TServerForm. DisconnectComboBoxUpdate;

var i:integer;

begin

ComboBox2. Clear;

for i:=0 to 44 do

begin

if DataSetForReport[i].Registered then ComboBox2. Items. Add (DataSetForReport[i].Name);

end;

end;

procedure TServerForm. CreateReport;

var

RangeW:word2000.range;

j:integer;

StrArr:array of string[30];

Data: WideString;

SData:string;

Sep, tmpRange, NumCols: OleVariant;

Parfs: Paragraphs;

Par: Paragraph;

begin

WordDocument1. Activate;

WordDocument1. Range. Font. Bold:=0;

WordDocument1. Range. Font. Size:=14;

WordDocument1. PageSetup. LeftMargin:=20;

WordDocument1. PageSetup. TopMargin:=20;

WordDocument1. PageSetup. RightMargin:=20;

WordDocument1. PageSetup. BottomMargin:=60;

SetLength (StrArr, ReportGrid. RowCount);

RangeW:=WordDocument1. Range (emptyParam, emptyParam);

tmpRange:=RangeW;

Parfs:=WordDocument1. Paragraphs;

par:=Parfs. Add(tmpRange);

tmpRange:=Par. Range.get\_end\_;

RangeW:=WordDocument1. Range(tmpRange);

SData:='';

Data:='ФИО@Группа@Дисциплина@Верных@Неверных@Время@Оценка@';

for j:=1 to ReportGrid. RowCount do

begin

begin // вывод информации по одному преподавателю

SData:=SData+ReportGrid. Cells [0, j]+'@'+ReportGrid. Cells [1, j]+'@'+ReportGrid. Cells [2, j]+'@'

+ReportGrid. Cells [4, j]+'@'+ReportGrid. Cells [5, j]+'@'+ReportGrid. Cells [6, j]+'@'+

ReportGrid. Cells [7, j]+'@';

Data:=Data+SData;

SData:='';

end;

end;

tmpRange:=RangeW;

Par:=Parfs. Add(tmpRange);

Par. Range. InsertBefore(Data);

Sep:='@';

NumCols:=7;

RangeW. ConvertToTableOld (Sep, EmptyParam, NumCols, EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam, EmptyParam);

WordDocument1. Disconnect;

SetLength (StrArr, 0);

end;

procedure TServerForm. Button1Click (Sender: TObject);

var

MsWord: Variant;

begin

try

MsWord:= CreateOleObject ('Word. Application');

MsWord. Visible:= True;

MsWord. Caption:='Отчет по реультатам тестирования';

CreateReport;

except

ShowMessage ('Невозможно запустить Microsoft Word');

Exit;

end;

end;

procedure TServerForm. SpeedButton1Click (Sender: TObject);

var Command:byte;

begin

if ComboBox2. ItemIndex>=0 then

begin

Command:=NM\_KickFromServer;

ServerSocket1. Socket. Connections [ComboBox2. ItemIndex].SendBuf (Command, 1);

end;

end;

procedure TServerForm. StringGrid1SelectCell (Sender: TObject; ACol,

ARow: Integer; var CanSelect: Boolean);

begin

SelectedRow:=ARow;

end;

procedure TServerForm. Button7Click (Sender: TObject);

begin

Memo1. Clear;

end;

procedure TServerForm. Button8Click (Sender: TObject);

begin

if SaveDialog1. Execute then Memo1. Lines. SaveToFile (SaveDialog1. FileName);

end;

procedure TServerForm. LogMessage (var Message: TMessage);

begin

Memo1. Lines. Add (DateTimeToStr(Now)+' '+PChar (Message.WParam));

end;

end.

unit QBaseWork;

interface

uses

Windows, Messages, SysUtils, Classes, Dialogs, IniFiles;

const

ErrWorkListLoad = 1;

ErrImputWorkNumberFault = 2;

ErrTeachersListLoad = 3;

ErrImputTeacherNumberFault = 4;

ErrQuestionsNotFound = 5;

ErrConfigIniFileWorkSetNotFound = 6;

ErrReadBuiletNumber = 7;

ErrQuestionWithInputedNumberNotFound = 8;

ErrQuestionFileWithInputedNumberNotFound = 9;

ErrInSelectedDirectoryNotQuestFileNameFound = 10;

ErrGenerationRndQuest = 11;

type

DBase=record

Works:HLringList;

Teachers:array of HLringList;

end;

type

TQuestDB = class

private

SelfParent:HWND;

NewBase:DBase;

WorksCount\_:integer;

WorkTimeLimit\_:String;

ProgRootDir:string;

ActiveWork:string;

ActiveTeacher:string;

ActiveWorkNum:byte;

ActiveTeacherNum:byte;

 ///////QUESTIONS /////////

ImgType:string;

QuestCount:integer;

QuestionsPathName:string;

ActivTransactionUser: String;

procedure ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL (ErrID:byte);

 ///////QUESTIONS /////////

function ConverHLrToIntNum (StringNum: string): integer;

function TestByDigit (DataString: string): boolean;

procedure SMessage (Message\_: string);

function UpdateQuestionsSet: boolean;

 // function GetWorkIndex (WorkName: string): integer;

 // function GetTeacherIndex (TeacherName: string): integer;

public

constructor Create (ParentHwnd:HWND);

destructor Destroy; override;

function SetActiveTeacher (Num: byte):boolean;

function SetActiveWork (Num: byte):boolean;

function GetWorksStringList:string;

function GetTeachersStringList:string;

property ActivWorkName:string read ActiveWork;

property ActivTeacherName:string read ActiveTeacher;

property TransactionUser:string read ActivTransactionUser write ActivTransactionUser;

property PubActivWorkNum:byte read ActiveWorkNum;

property PubActivTeacherNum:byte read ActiveTeacherNum;

property QuestionsFullPath:string read QuestionsPathName;

function GetWorkByIndex (i: byte): string;

function GetTeacherByIndex (i: byte): string;

 ///////QUESTIONS /////////

property ImgFileType:string read ImgType;

property QuestionsCount:integer read QuestCount;

property WorkTimeLimit: String read WorkTimeLimit\_;

function GetBuiletByNum (Num: integer): string;

function GetFileBuiletByNumBuilet (BuiletNum, FileNum: integer): string;

function GetRandomFileBuilet (BuiletNum: integer): string;

function GetTrueAnswerForBuilet (QuestionPath: string): integer;

function SetTrueAnswerForBuilet (QuestionPath: string; TrueAnswer: Integer): boolean;

end;

implementation

{TQuestDB}

constructor TQuestDB. Create (ParentHwnd:HWND);

var ExeName:PChar;

AppName: String;

ExeNameLen:byte;

 /////

NewSearch\_:TSearchRec;

i, ii:byte;

QuestionPathName:string;

QCount:integer;

FOptions:TIniFile;

begin

SelfParent:=ParentHwnd;

GetMem (ExeName, 255);

ExeNameLen:=255;

GetModuleFileName (0, ExeName, ExeNameLen); // определяем имя исполняемого модуля

AppName:=StrPas(ExeName);

ProgRootDir:=ExtractFileDir(AppName);

WorksCount\_:=0;

NewBase. Works:=HLringList. Create; // заполняем список работ

FindFirst (ProgRootDir+'\Questions\\*', faDirectory, NewSearch\_);

repeat

if NewSearch\_.Name[1]<>'.' then

begin

NewBase. Works. Add (NewSearch\_.Name);

inc (WorksCount\_);

end;

until FindNext (NewSearch\_)<>0;

FindClose (NewSearch\_);

 // Заполняем списки преподов

SetLength (NewBase. Teachers, WorksCount\_);

for i:=0 to WorksCount\_-1 do

begin

NewBase. Teachers[i]:=HLringList. Create;

FindFirst (ProgRootDir+'\Questions\'+NewBase. Works. Strings[i]+'\\*', faDirectory, NewSearch\_);

repeat

if NewSearch\_.Name[1]<>'.' then NewBase. Teachers[i].Add (NewSearch\_.Name);

until FindNext (NewSearch\_)<>0;

FindClose (NewSearch\_);

end;

for i:=0 to NewBase. Works. Count-1 do

begin

for ii:=0 to NewBase. Teachers[i].Count-1 do

begin

QuestionPathName:=ProgRootDir+'\Questions\'+NewBase. Works. Strings[i]+'\'+ NewBase. Teachers[i].Strings[ii];

if FileExists (QuestionPathName+'\WorkSet.ini') then

begin

FOptions:=TIniFile. Create (QuestionPathName+'\WorkSet.ini');

QCount:=0;

FindFirst (QuestionPathName+'\\*', faDirectory, NewSearch\_);

repeat

if NewSearch\_.Name[1]<>'.' then

if TestByDigit (NewSearch\_.Name) then inc(QCount);

until FindNext (NewSearch\_)<>0;

FindClose (NewSearch\_);

FOptions. WriteInteger ('QuestionCount', 'value', QCount);

FOptions. Free;

if QCount>0 then QuestCount:=QCount else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrQuestionsNotFound);

end else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrConfigIniFileWorkSetNotFound);

end;

end;

end;

destructor TQuestDB. Destroy;

var i:integer;

begin

for i:=0 to NewBase. Works. Count-1 do

begin

NewBase. Teachers[i].Destroy;

end;

SetLength (NewBase. Teachers, 0);

NewBase. Works. Destroy;

inherited;

end;

function TQuestDB. SetActiveWork (Num:byte):boolean;

begin

result:=false;

if Num<NewBase. Works. Count then

begin

ActiveWork:=NewBase. Works. Strings[Num];

ActiveWorkNum:=Num;

result:=true;

end else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrImputWorkNumberFault);

end;

function TQuestDB. SetActiveTeacher (Num:byte):boolean;

begin

result:=false;

if Num<NewBase. Teachers[ActiveWorkNum].Count then

begin

ActiveTeacher:=NewBase. Teachers[ActiveWorkNum].Strings[Num];

ActiveTeacherNum:=Num;

if UpdateQuestionsSet then result:=true;

end else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrImputTeacherNumberFault);

end;

function TQuestDB. GetTeachersStringList: string;

var i:integer;

begin

Result:='';

for i:=0 to NewBase. Teachers[ActiveWorkNum].Count-1 do Result:=Result+NewBase. Teachers[ActiveWorkNum].Strings[i]+'|';

Result:=Result+'>';

end;

function TQuestDB. GetWorksStringList: string;

var i:integer;

begin

Result:='';

for i:=0 to NewBase. Works. Count-1 do Result:=Result+NewBase. Works. Strings[i]+'|';

Result:=Result+'>';

end;

function TQuestDB. GetWorkByIndex (i:byte): string;

begin

if i<=NewBase. Works. Count-1 then Result:=NewBase. Works. Strings[i] else Result:='';

end;

function TQuestDB. GetTeacherByIndex (i:byte): string;

begin

if i<=NewBase. Teachers[ActiveWorkNum].Count-1 then

Result:=NewBase. Teachers[ActiveWorkNum].Strings[i] else

Result:='';

end;

procedure TQuestDB.ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL (ErrID: byte);

begin

Case ErrID of

ErrWorkListLoad:

begin

SMessage ('Base read works error');

end;

ErrTeachersListLoad:

begin

SMessage ('Base read teachers error');

end;

ErrImputWorkNumberFault:

SMessage ('Imput work number fault');

ErrImputTeacherNumberFault:

SMessage ('Imput work number fault');

ErrQuestionsNotFound:

SMessage ('No questions found in base');

ErrConfigIniFileWorkSetNotFound:

SMessage ('Config file WorkSet.ini not found');

ErrReadBuiletNumber:

SMessage ('Error with read number of builet');

ErrQuestionWithInputedNumberNotFound:

SMessage ('Direstory with inputed number (QuestionNum) is not found (number out of range)');

ErrQuestionFileWithInputedNumberNotFound:

SMessage ('File with inputed number (QuestionName) is not found (number out of range)');

ErrInSelectedDirectoryNotQuestFileNameFound:

SMessage ('In the selected tirectory question file is not found');

ErrGenerationRndQuest:

SMessage ('Error by generation random question file maybe question directory is not found');

ErrInvalidFileNameTraslate:

SMessage ('Invalid Translate question name filename STR to INT maybe filename error');

end;

end;

Procedure TQuestDB.SMessage (Message\_:string);

begin

SendMessage (SelfParent, WM\_User+2, DWord (PChar(TransactionUser+' '+Message\_)), 0);

end;

 /////////////////QUESTIONS ////////////////

function TQuestDB. UpdateQuestionsSet:boolean;

var QCount:integer;

EnumFileDir:TSearchRec;

FOptions:TIniFile;

TryConvert:TDateTime;

WorkTimeLim:string;

begin

QuestionsPathName:=ProgRootDir+'\Questions\'+ActiveWork+'\'+ActiveTeacher;

try

try

FOptions:=TIniFile. Create (QuestionsPathName+'\WorkSet.ini');

QuestCount:=FOptions. ReadInteger ('QuestionCount', 'value', – 1);

WorkTimeLim:=FOptions. ReadString ('TimeForWork', 'value', '0:00:00');

TryConvert:=StrToTime(WorkTimeLim);

WorkTimeLimit\_:=WorkTimeLim;

ImgType:=FOptions. ReadString ('ImgType', 'value', 'bmp');

FOptions. Destroy;

finally

if QuestCount>0 then result:=true else result:=false;

end;

except

result:=false;

end;

end;

function TQuestDB. ConverHLrToIntNum (StringNum:string):integer;

var ProtectAssign:integer;

begin

if TestByDigit(StringNum) then

begin

ProtectAssign:=StrToInt(StringNum);

result:=ProtectAssign;

end else

begin

ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrReadBuiletNumber);

result:=-1;

end;

end;

function TQuestDB. TestByDigit (DataString:string):boolean;

var DataLen:byte;

Offs:byte;

begin

Result:=true;

DataLen:=Length(DataString);

for Offs:=1 to DataLen do

if not (DataString[Offs] in ['0'..'9']) then

begin

result:=false;

break;

end;

end;

function TQuestDB. GetBuiletByNum (Num:integer):string;

var EnumBuiletsFile:TSearchRec;

StringBuiletNum:string;

begin

Result:='';

FindFirst (QuestionsPathName+'\\*', faDirectory, EnumBuiletsFile);

repeat

if EnumBuiletsFile. Name[1]<>'.' then

begin

StringBuiletNum:=EnumBuiletsFile. Name;

if TestByDigit(StringBuiletNum) then

if ConverHLrToIntNum(StringBuiletNum)=Num then

begin

result:=QuestionsPathName+'\'+EnumBuiletsFile. Name;

break;

end;

end;

until FindNext(EnumBuiletsFile)<>0;

FindClose(EnumBuiletsFile);

If Result='' then ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrQuestionWithInputedNumberNotFound);

end;

function TQuestDB. GetFileBuiletByNumBuilet (BuiletNum, FileNum:integer):string;

var EnumBuiletsNamesFile:TSearchRec;

StringBuiletNum:string;

begin

Result:='';

FindFirst (QuestionsPathName+'\'+IntToStr(BuiletNum)+'\\*', faAnyFile, EnumBuiletsNamesFile);

repeat

if EnumBuiletsNamesFile. Name[1]<>'.' then

begin

StringBuiletNum:=EnumBuiletsNamesFile. Name;

Delete (StringBuiletNum, Length(StringBuiletNum) – 3,4);

if TestByDigit(StringBuiletNum) then

if ConverHLrToIntNum(StringBuiletNum)=FileNum then

begin

result:=QuestionsPathName+'\'+EnumBuiletsNamesFile. Name;

break;

end;

end;

until FindNext(EnumBuiletsNamesFile)<>0;

FindClose(EnumBuiletsNamesFile);

If Result='' then ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrQuestionFileWithInputedNumberNotFound);

end;

function TQuestDB. GetRandomFileBuilet (BuiletNum:integer):string;

var EnumBuiletsNamesFile:TSearchRec;

RndCount:integer;

FileList:HLringList;

WorkPath:string;

begin

Result:='';

FileList:=HLringList. Create;

FileList. Clear;

WorkPath:=QuestionsPathName+'\'+IntToStr(BuiletNum);

if DirectoryExists(WorkPath) then

begin

FindFirst (WorkPath+'\\*', faAnyFile, EnumBuiletsNamesFile);

repeat

if EnumBuiletsNamesFile. Name[1]<>'.' then

FileList. Add (EnumBuiletsNamesFile. Name);

until FindNext(EnumBuiletsNamesFile)<>0;

FindClose(EnumBuiletsNamesFile);

if FileList. Count>0 then

begin

Randomize;

RndCount:=Random (FileList. Count);

Result:=QuestionsPathName+'\'+IntToStr(BuiletNum)+'\'+FileList. Strings[RndCount];

end;

end;

FileList. Destroy;

If Result='' then ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrGenerationRndQuest);

end;

function TQuestDB. GetTrueAnswerForBuilet (QuestionPath:string):integer;

var QuestNum:integer;

TmpStr:string;

KeyFilePath:string;

TempQuestionsList:HLringList;

begin

Result:=-1;

QuestNum:=0;

TmpStr:=ExtractFileName(QuestionPath);

Delete (TmpStr, Length(TmpStr) – Length (ExtractFileExt(TmpStr))+1, Length (ExtractFileExt(TmpStr)));

if (TestByDigit(TmpStr)) and (Length(TmpStr)<5) then

begin

QuestNum:=StrToInt(TmpStr);

end else

begin

ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrInvalidFileNameTraslate);

Result:=-1;

exit;

end;

KeyFilePath:=ExtractFilePath (ExtractFileDir(QuestionPath))+'QuestKey.ini';

if FileExists(KeyFilePath) then

begin

TempQuestionsList:=HLringList. Create;

TempQuestionsList. LoadFromFile(KeyFilePath);

Result:=StrToInt (TempQuestionsList. Strings[QuestNum]);

TempQuestionsList. Destroy;

end else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrConfigIniFileWorkSetNotFound);

end;

function TQuestDB. SetTrueAnswerForBuilet (QuestionPath:string; TrueAnswer: Integer):boolean;

var QuestNum:integer;

TmpStr:string;

KeyFilePath:string;

TempQuestionsList:HLringList;

begin

Result:=false;

QuestNum:=0;

TmpStr:=ExtractFileName(QuestionPath);

Delete (TmpStr, Length(TmpStr) – Length (ExtractFileExt(TmpStr))+1, Length (ExtractFileExt(TmpStr)));

if (TestByDigit(TmpStr)) and (Length(TmpStr)<5) then

begin

QuestNum:=StrToInt(TmpStr);

end else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrInvalidFileNameTraslate);

KeyFilePath:=ExtractFilePath (ExtractFileDir(QuestionPath))+'QuestKey.ini';

if FileExists(KeyFilePath) then

begin

TempQuestionsList:=HLringList. Create;

TempQuestionsList. LoadFromFile(KeyFilePath);

TempQuestionsList. Strings[QuestNum]:=IntToStr(TrueAnswer);

TempQuestionsList. SaveToFile (KeyFilePath+'\_');

TempQuestionsList. Destroy;

DeleteFile(KeyFilePath);

RenameFile (KeyFilePath+'\_', KeyFilePath);

Result:=true;

end else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrConfigIniFileWorkSetNotFound);

end;

end.

unit UBaseWork;

interface

uses Windows, Messages, SysUtils, Classes, Dialogs, IniFiles;

const

ErrImputGroupNumberFault = 1;

ErrImputUserNumberFault = 2;

type

UsersDBase=record

Groups:HLringList;

Users:array of HLringList;

end;

type

TUsersDB = class

private

SelfParent:HWND;

UsersDataBase: UsersDBase;

GroupsCount:integer;

ProgRootDir:string;

ActiveGroup:string;

ActiveUser:string;

ActivStationIP:string;

ActiveGroupNum:byte;

ActiveUserNum:byte;

procedure ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL (ErrID: byte);

procedure SMessage (Message\_: string);

public

property TransactionIP:string read ActivStationIP write ActivStationIP;

property ActiveUserName:string read ActiveUser;

property ActiveGroupName:string read ActiveGroup;

function SetActiveGroup (Num: byte): boolean;

function SetActiveUser (Num: byte): boolean;

function GetGroupByIndex (i: byte): string;

function GetUserByIndex (i: byte): string;

function GetGroupsStringList: string;

function GetUsersStringList: string;

constructor Create (ParentHwnd:HWND);

destructor Destroy; override;

end;

implementation

{TQuestDB}

constructor TUsersDB. Create (ParentHwnd: HWND);

var ExeName:PChar;

AppName: String;

ExeNameLen:byte;

 /////

NewSearch\_:TSearchRec;

CleanName:string;

i:byte;

begin

SelfParent:=ParentHwnd;

GetMem (ExeName, 255);

ExeNameLen:=255;

GetModuleFileName (0, ExeName, ExeNameLen); // определяем имя исполняемого модуля

AppName:=StrPas(ExeName);

ProgRootDir:=ExtractFileDir(AppName);

GroupsCount:=0;

UsersDataBase. Groups:=HLringList. Create;

FindFirst (ProgRootDir+'\Groups\\*', faDirectory, NewSearch\_);

repeat

if NewSearch\_.Name[1]<>'.' then

begin

UsersDataBase. Groups. Add (NewSearch\_.Name);

inc(GroupsCount);

end;

until FindNext (NewSearch\_)<>0;

FindClose (NewSearch\_);

SetLength (UsersDataBase. Users, GroupsCount);

for i:=0 to GroupsCount-1 do

begin

UsersDataBase. Users[i]:=HLringList. Create;

UsersDataBase. Users[i].LoadFromFile (ProgRootDir+'\Groups\'+UsersDataBase. Groups. Strings[i]);

CleanName:=UsersDataBase. Groups. Strings[i];

Delete (CleanName, Length(CleanName) – 3,4);

UsersDataBase. Groups. Strings[i]:=CleanName;

end;

end;

destructor TUsersDB. Destroy;

var i:integer;

begin

for i:=0 to UsersDataBase. Groups. Count-1 do

begin

UsersDataBase. Users[i].Destroy;

end;

SetLength (UsersDataBase. Users, 0);

UsersDataBase. Groups. Destroy;

inherited;

end;

function TUsersDB. SetActiveGroup (Num:byte):boolean;

begin

result:=false;

if Num< UsersDataBase. Groups. Count then

begin

ActiveGroup:=UsersDataBase. Groups. Strings[Num];

ActiveGroupNum:=Num;

result:=true;

end else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrImputGroupNumberFault);

end;

function TUsersDB. SetActiveUser (Num:byte):boolean;

begin

result:=false;

if Num< UsersDataBase. Users[ActiveGroupNum].Count then

begin

ActiveUser:=UsersDataBase. Users[ActiveGroupNum].Strings[num];

ActiveUserNum:=Num;

result:=true;

end else ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL(ErrImputUserNumberFault);

end;

procedure TUsersDB.ERROR\_MESSAGE\_FOR\_DEBUG\_LEVEL (ErrID: byte);

begin

Case ErrID of

ErrImputGroupNumberFault:

SMessage ('Imput group number fault');

ErrImputUserNumberFault:

SMessage ('Imput user number fault');

end;

end;

Procedure TUsersDB.SMessage (Message\_:string);

begin

SendMessage (SelfParent, WM\_User+2, DWord (PChar(ActivStationIP+' '+Message\_)), 0);

end;

function TUsersDB. GetGroupByIndex (i:byte): string;

begin

if i<=UsersDataBase. Groups. Count-1 then Result:=UsersDataBase. Groups. Strings[i] else Result:='';

end;

function TUsersDB. GetUserByIndex (i:byte): string;

begin

if i<=UsersDataBase. Users[ActiveGroupNum].Count-1 then

Result:=UsersDataBase. Users[ActiveGroupNum].Strings[i] else Result:='';

end;

function TUsersDB. GetGroupsStringList: string;

var i:integer;

begin

Result:='';

for i:=0 to UsersDataBase. Groups. Count-1 do Result:=Result+UsersDataBase. Groups. Strings[i]+'|';

Result:=Result+'>';

end;

function TUsersDB. GetUsersStringList: string;

var i:integer;

begin

Result:='';

for i:=0 to UsersDataBase. Users[ActiveGroupNum].Count-1 do Result:=Result+UsersDataBase. Users[ActiveGroupNum].Strings[i]+'|';

Result:=Result+'>';

end;

end.

# Приложение 2

# Листинг кода клиентской части программы

unit Registation;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls;

type

HLartForm = class(TForm)

Panel2: TPanel;

ComboBox3: TComboBox;

ComboBox4: TComboBox;

Label5: TLabel;

Label6: TLabel;

Bevel2: TBevel;

Bevel3: TBevel;

Panel1: TPanel;

Bevel4: TBevel;

Bevel5: TBevel;

Label3: TLabel;

Label4: TLabel;

ComboBox1: TComboBox;

ComboBox2: TComboBox;

Bevel6: TBevel;

Bevel7: TBevel;

Panel3: TPanel;

Bevel1: TBevel;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Panel4: TPanel;

procedure ComboBox1Change (Sender: TObject);

procedure Button2Click (Sender: TObject);

procedure Button1Click (Sender: TObject);

procedure Button3Click (Sender: TObject);

procedure ComboBox3Change (Sender: TObject);

procedure ComboBox2Change (Sender: TObject);

procedure FormClose (Sender: TObject; var Action: TCloseAction);

private

ServerIPAddress:string[15]; //IP адрес

Steps:byte; // номер шага регистрации (условно)

NoModify:boolean; // триггер интерфейса

function ReadServerIP: string; // чтение из файла IP.DAT информации о IP адресе сервера

public

procedure GetConnect; // Установка соединение

procedure HideWin\_(YN: boolean); // скрыть элементы управления Windows (TaskBar, Deskdop)

procedure ExitProgram;

end;

var

StartForm: HLartForm;

implementation

uses MainForm;

{ /////////////////////////////////////////////////////

BEGIN

Сервисные подпрограммы

 ////////////////////////////////////////////////////// }

function HLartForm. ReadServerIP: string;

var IPInfFile:textfile;

IP:string;

begin

if fileexists (extractfilepath(application. ExeName)+'IP. Dat') then

begin

assignfile (IPInfFile, extractfilepath (application. ExeName)+'IP. Dat');

{$i-}

reset(IPInfFile);

Readln (IPInfFile, IP);

closefile(IPInfFile);

{$i+}

if ip<>'' then

begin

ReadServerIP:=IP;

end

else ReadServerIP:='127.0.0.1';

end else

begin

ReadServerIP:='127.0.0.1';

end;

end;

procedure HLartForm. HideWin\_(YN:boolean);

var Wnd: hWnd;

ClassName:PChar;

ClassNameLen:byte;

Res:string;

begin

Wnd:=FindWindow ('Progman', 'Program Manager');

while wnd<>0 do

begin

wnd:=GetWindow (Wnd, GW\_CHILD);

ClassNameLen:=0;

GetClassName (Wnd, ClassName, ClassNameLen);

SeHLring (Res, ClassName, ClassNameLen);

SeHLring (Res, ClassName, StrLen(ClassName));

if Res='SysListView32' then

begin

if YN=true then

begin

ShowWindow (Wnd, SW\_Hide);

ShowWindow (findwINDOW('Shell\_TrayWnd', nil), SW\_Hide);

end else

begin

ShowWindow (Wnd, SW\_Show);

ShowWindow (findwINDOW('Shell\_TrayWnd', nil), SW\_Show);

end;

break;

end;

end;

FreeMem (ClassName, 255);

end;

procedure HLartForm. ExitProgram;

begin

HideWin\_(false);

Application. ProcessMessages;

Application. Terminate;

end;

{ /////////////////////////////////////////////////////

Сервисные подпрограммы

END

 ////////////////////////////////////////////////////// }

{ /////////////////////////////////////////////////////

BEGIN

Сетевые подпрограммы

 ////////////////////////////////////////////////////// }

procedure HLartForm. GetConnect;

begin

try

ServerIPAddress:=ReadServerIP;

TestForm. TestSocket. Address:=ServerIPAddress;

TestForm. TestSocket. Active:=true;

except

end;

end;

{ /////////////////////////////////////////////////////

Сетевые подпрограммы

END

 ////////////////////////////////////////////////////// }

{ /////////////////////////////////////////////////////

BEGIN

Обработка пользовательского интерфейса

 ////////////////////////////////////////////////////// }

procedure HLartForm. ComboBox1Change (Sender: TObject);

var Data:string;

begin

Data:=Char (NM\_Register2)+Char (TestForm. MyNumber)+Char (ComboBox1. ItemIndex);

TestForm. TestSocket. Socket. SendBuf (Pointer(Data)^, Length(Data));

ComboBox3. Clear;

ComboBox4. Clear;

ComboBox2. Clear;

NoModify:=false;

Steps:=0;

end;

procedure HLartForm. Button2Click (Sender: TObject);

begin

Close;

end;

procedure HLartForm. Button1Click (Sender: TObject);

var Data:string;

begin

case Steps of // Дальнейшее действие

0:if ComboBox2. Text<>'' then

begin

NoModify:=true;

Data:=Char (NM\_RegisterGetWorks)+Char (TestForm. MyNumber)+Char (ComboBox1. ItemIndex);

TestForm. TestSocket. Socket. SendBuf (Pointer(Data)^, Length(Data)); // Запрос на получение списка предметов

end;

Button3. Enabled:=true;

Panel1. Hide;

Panel2. Show; Steps:=1;

end;

1: if Panel2. Visible then

begin

if ComboBox4. Text<>'' then

begin

Data:=Char (NM\_RegisterOK)+Char (TestForm. MyNumber)+

Char (ComboBox1. ItemIndex)+Char (ComboBox2. ItemIndex)+Char (ComboBox3. ItemIndex)+Char (ComboBox4. ItemIndex);

TestForm. TestSocket. Socket. SendBuf (Pointer(Data)^, Length(Data)); // Отсылка сведений для

 // окончательной регистрации

Self. Hide;

HideWin\_(true);

end;

end else

begin

Panel1. Hide;

Panel2. Show;

Button3. Enabled:=true;

Steps:=1;

end;

end;

end;

procedure HLartForm. Button3Click (Sender: TObject);

begin

Panel2. Hide;

Panel1. Show;

Button3. Enabled:=false;

end;

procedure HLartForm. ComboBox3Change (Sender: TObject);

var Data:string;

begin

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, WinSock, ExtCtrls, Buttons, StdCtrls, ScktComp;

const

NM\_Register1 = 6; // прием списка групп

NM\_Register2 = 7; // запрос на список студентов

NM\_RegisterGetWorks = 66; // запрос / ответ 'список предметов'

NM\_RegisterGetTeachers = 77; // запрос / ответ 'список преподователей'

NM\_RegisterOK = 8; // клиент зарегистрирован

NM\_Service = 31; // прием сервисной информации

NM\_TestEvent = 55; // событие по ходу тестирования

NM\_FileOperation = 10; // сетевая операция с файлами

NM\_EndOfTest = 33; // окончание тестирования

NM\_KickFromServer = 44; // отключение от сервера администратором

NM\_Wait = 61;

NM\_DataError = 54; // проблема с БД

procedure TTestForm. TestSocketRead (Sender: TObject;

Socket: TCustomWinSocket);

type TDataBuffer=array of byte; // буфер данных

var Data, Data1:string; // данные

SendLen:integer;

DataBuffer:TDataBuffer;

i: Word;

Command:byte;

GetSize:PInt64;

SizeOfFilename:byte;

PTrueAnswer:PWord;

PTimeForPassTest:PDouble;

begin

SendLen:=Socket. ReceiveLength; // размер принятых данных

SetLength (DataBuffer, SendLen);

Socket. ReceiveBuf (Pointer(DataBuffer)^, SendLen); // заполняем буфер

if lock then // если в режиме приема файла то продолжить прием

begin

MakePointer:=DWORD(DataBuffer);

NewFile. WriteBuffer (Pointer(MakePointer)^, SendLen);

SendedSize:=SendedSize+SendLen;

if SendedSize=FileSize then // если приняли весь файл то выход

begin

lock:=false;

NewFile. Destroy;

SetImg(FileName);

end;

end else

begin

Command:=DataBuffer[0];

case Command of

NM\_Register1:

begin

MyNumber:=DataBuffer[1];

i:=2;

while i<=SendLen-3 do

begin

Data:='';

while DataBuffer[i]<>byte ('|') do

begin

Data:=Data+Char (DataBuffer[i]);

inc(i);

end;

if Data<>'' then StartForm. ComboBox1. Items. Add(Data);

if DataBuffer [i+1]=byte ('>') then break;

inc(i);

end;

end;

NM\_Register2: // список студентов

begin

i:=1;

while i<=SendLen-2 do

begin

Data:='';

while DataBuffer[i]<>byte ('|') do

begin

Data:=Data+Char (DataBuffer[i]);

inc(i);

end;

if Data<>'' then StartForm. ComboBox2. Items. Add(Data);

if DataBuffer [i+1]=byte ('>') then break;

inc(i);

end;

end;

NM\_RegisterGetWorks:

begin

i:=1;

StartForm. ComboBox3. Clear;

while i<=SendLen-2 do

begin

Data:='';

while DataBuffer[i]<>byte ('|') do

begin

Data:=Data+Char (DataBuffer[i]);

inc(i);

end;

if Data<>'' then StartForm. ComboBox3. Items. Add(Data);

if DataBuffer [i+1]=byte ('>') then break;

inc(i);

end;

end;

NM\_RegisterGetTeachers:

begin

StartForm. ComboBox4. Clear;

i:=1;

while i<=SendLen-2 do

begin

Data:='';

while DataBuffer[i]<>byte ('|') do

begin

Data:=Data+Char (DataBuffer[i]);

inc(i);

end;

if Data<>'' then StartForm. ComboBox4. Items. Add(Data);

if DataBuffer [i+1]=byte ('>') then break;

inc(i);

end;

end;

NM\_FileOperation:

begin

lock:=true;

PTrueAnswer:=Addr (DataBuffer[1]);

TrueAnswer:=PTrueAnswer^;

QuestionStyle:=DataBuffer[3];

GetSize:=Addr (DataBuffer[4]);

FileSize:=GetSize^;

SizeOfFilename:=DataBuffer[12];

Filename:=ApplicationPath+'Data.tmp'; // имя передаваемого файла

Deletefile(FileName);

NewFile:=TFileStream. Create (FileName, fmCreate);

NewFile. Position:=0;

MakePointer:=DWORD(DataBuffer)+13+SizeOfFilename; // 13=1+1+1+1+8+1

NewFile. WriteBuffer (Pointer(MakePointer)^, SendLen-13-SizeOfFilename);

SendedSize:=SendLen-13-SizeOfFilename;

if SendedSize=FileSize then // если приняли весь файл то выход

begin

lock:=false;

NewFile. Destroy;

SetImg(FileName);

end;

end;

NM\_EndOfTest:

begin

SpeedButton5. Enabled:=false;

TestPassed:=true;

Mark:=DataBuffer[1];

PostMessage (Handle, WM\_User, 0,0);

end;

NM\_KickFromServer:

begin

TestTerminated:=true;

Label7. Hide;

Label8. Hide;

Button2. Hide;

Panel7. Caption:='Тестирование прервано';

PostMessage (Handle, WM\_User, 0,0);

end;

NM\_Service:

begin

QuestionsCount:=DataBuffer[1];

PTimeForPassTest:=Addr (DataBuffer[2]);

TimeForPassTest:=TTime (PTimeForPassTest^);

end;

NM\_DataError:

begin

SendLen:=DataBuffer[1];

Data1:=Copy (PChar(DataBuffer), 3, SendLen)+#13+#10+#0;

PostMessage (Handle, WM\_User+1, DWORD (PChar(Data1)), 1);

end;

NM\_Wait: ShowMessage('Wait');

end;

end;

SetLength (DataBuffer, 0);

end;

procedure TTestForm. CloseNetworkSocket (var Message: TMessage);

begin

TestSocket. Active:=false;

TestSocket.close;

if TestForm. Visible then

begin

Panel8. Hide;

Panel7. Top:=Panel8. Top;

Panel7. Left:=Panel8. Left;

Panel7. Width:=Panel8. Width;

Panel7. Height:=Panel8. Height;

Panel7. Visible:=true;

if TestPassed then Panel7. Caption:=IntToStr(Mark) else

begin

Application. ProcessMessages;

Sleep(4000);

Application. ProcessMessages;

Application. Terminate;

end;

end else // если окно теста не открыто

begin

StartForm. Panel4. Visible:=true;

Application. ProcessMessages;

Sleep(4000);

Application. ProcessMessages;

Application. Terminate;

end;

end;

procedure TTestForm. TestSocketDisconnect (Sender: TObject;

Socket: TCustomWinSocket);

begin

if not (TestPassed or TestTerminated) then Application. Terminate;

end;

{ /////////////////////////////////////////////////////

Сетевые подпрограммы

END

 ////////////////////////////////////////////////////// }

end;

end.

# Литература

1. Архангельский А.Я. Delphi 7 Справочное пособие. – М., Бином-Пресс. -2004. -1024 с.
2. Архангельский А.Я. Программирование в Delphi 7 + дискета, Бином, 2005
3. Бондаренко Е.А. Технические средства обучения в современной школе, Юверс, 2004
4. Вигерс Карл. Разработка требований к программному обеспечению. /Пер, с англ. – М.: Издательско-торговый дом «Русская Редакция», 2004. - 576 с.
5. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб.: Питер, 2001. – 384 с.: ил.
6. Глушаков С.В., Клевцов А.Л., Программирование в среде Delphi 7.0, Фолио 2003
7. Дьяконов В.П. Новые информационные технологии, Солон-Пресс, 2005
8. Земсков А.И., Шрайберг Я.Л. Электронные библиотеки, Либерея, 2003
9. Клименко Р.Н. Оптимизация и автоматизация работы на ПК на 100% (+CD), Питер Пресс, 2007
10. Колин К.К. Фундаментальные основы информатики: социальная информатика / Учебное пособие для вузов. – М.: Академический проект, 200 –350 с.
11. Кондратьев Г.Г. Осваиваем Windows XP, Питер, 2005
12. Коплиен Дж., Мультипарадигменное проектирование для C++, Питер, 2005
13. Красильникова В.А. Становление и развитие компьютерных технологий обучения: Монография. – М.: ИИО РАО, 2002. – 168 с.
14. Круглински Д., Уингоу С, Шеферд Дж. Программирование на Microsoft Visual C++ 6.0 для профессионалов. /Пер, с англ. – СПб: Питер; М.: Издательско-торговый дом «Русская Редакция», 2004. – 861 с.
15. Леонтьев Б.К., Мультимедия Microsoft Windows без страха, Новый издательский дом, 2005
16. Мандел Т. Дизайн интерфейсов, ДМК, 2005
17. Музыченко Е.В., Фролов И.Б., Мультимедия для Windows, 2003
18. Пайс А. Гении науки. – М.: Институт компьютерных исследований, 2002
19. Архангельский А.А. Программирование в Delphi. – М.: Бином, 2003. – 1231 с.
20. Гофман В.Э., Хомоненко А.Д. Delphi 5. – СПб.: БХВ – Санкт Петербург, 2000. – 800 с.
21. Епанешников А., Епанешников В. Программирование в среде Delphi: Учебное пособие: В 4-х ч. Ч. 4. Работа с базами данных. Организация справочной системы – М.: ДИАЛОГ – МИФИ, 1998. – 400 с.
22. Зубков Сергей Владимирович Assembler для Dos, Windows, Unix. – М.: ДМКПресс, 2000. – 652 с.
23. Кэнту Марко Delphi 5.0 для профессионалов. – СПб.: Питер, 2001. – 1064 с.
24. Пирогов В.Ю. Assembler учебный курс. – М.: «Нолидж», 2001. – 926 с.
25. Рейнхардт Р., Ленц Д.У. Flash 5. Библия пользователя. – М.: «Вильямс», 2001. – 1164 с.
26. Фигурнов В.Э. IBM PC для пользователя. Изд. 7-е, перераб. и доп. – М.: ИНФРА – М, 1998. – 640 с.
27. Батищев П.С. Электронный On-Line учебник по курсу информатика.
28. Ивановский Р.И. Компьютерные технологии в науке и образовании. Практика применения систем Math CAD Pro, Высшая школа, 2003
29. Каймин В.А., Жданов В.С. и др. «Информатика» для поступающих в ВУЗы. Москва, АСТ, 2006 г.
30. Кудрявцев Е.М. Оформление дипломного проекта на компьютере, АСВ, 2004