**ВВЕДЕНИЕ**

В современном мире стремительными темпами развиваются информационные технологии и те сферы человеческой деятельности, которые с ними связаны. С каждым годом всё больший и больший объём информации обрабатывается при помощи современных и продолжающих совершенствоваться персональных компьютерах.

Любая организация нуждается в своевременном доступе к информации. Ценность информации в современном мире очень высока. Роль распорядителей информации в современном мире чаще всего выполняют базы данных. Базы данных обеспечивают надежное хранение информации, структурированном виде и своевременный доступ к ней. Практически любая современная организация нуждается в базе данных, удовлетворяющей те или иные потребности по хранению, управлению и администрированию данных.

Главная цель, которая стоит перед написанием курсового проекта – это приобретение практических навыков проектирования базы данных «Центр детского творчества» с использованием программ СУБД. В ходе выполнения данного курсового проекта необходимо будет разработать информационную базу данных и программное обеспечение для работы совместно с базой данных для центра детского творчества, которые помогут пользователю легко найти нужную информацию о сотруднике, детях, группах и событиях в любом момент времени.

Пояснительная записка к курсовому проекту состоит из шести разделов.

В разделе «Постановка задачи» исследуется предметная область, определяется круг задач, которые должны быть автоматизированы в программном приложении, задаются функциональные требования.

Во втором разделе «Вычислительная система» описываются используемые технические средства, требования к аппаратному обеспечению и конфигурации, характеристика операционной системы.

В разделе «Проектирование задачи» описываются математическая модель решения алгоритма, классы, разработанные функции и элементы управления функциями.

В четвертом разделе «Описание программного средства» изложены общие сведения программного средства, функциональное назначение.

В пятом разделе «Методика испытаний» описываются требования к техническим средствам, к характеристикам программы, представлено полное тестирование программы.

В последнем разделе «Применение» описывается назначение программы, условия применения и справочная система.

Заключение состоит из описания и степени соответствия поставленной задачи.

Приложение включает в себя программный код с необходимыми комментариями.

# 1. ПОСТАНОВКА ЗАДАЧИ

## **1.1 Организационная сущность задачи**

Целью данного курсового проекта является реализация программной системы «Цент детского творчества» для автоматизации деятельности и обеспечения хранения, накопления и предоставления информации

Задачи проекта:

* Анализ предметной области;
* Построение инфологической (концептуальной) модели предметной области;
* Разработка логической структуры базы данных;
* Разработка физической структуры базы данных;
* Организация базы данных;
* Разграничение прав доступа.
* Разработка пользовательского интерфейса;
* Реализация проекта в конкретной среде программирования;
* Наполнение базы данных;

## **1.2 Информационная модель**

Наиболее распространенным средством моделирования данных являются диаграммы UML. С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи).

Каждая сущность должна обладать уникальным идентификатором и некоторыми свойствами:

- иметь уникальное имя, и к одному и тому же имени должна всегда применяться одна и та же интерпретация. Одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;

- обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;

- обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности;

- обладать любым количеством связей с другими сущностями модели.

Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности.

Связь - это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности родителя.

Атрибут - любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств, ассоциированных со множеством реальных или абстрактных объектов.

Экземпляр атрибута - это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Выделим базовые сущности данного курсового проекта:

• Дети. Атрибуты: информация о детях, родители.

• События. Атрибуты – описание, дата проведения, заметки, цена.

• Группы. Атрибуты – сотрудник, класс, расписание, размер группы, цена.

• Сотрудники. Атрибуты – ФИО, дата рождения, квалификация.

Родители и Информацию о детях будем рассматривать как связь с детьми. Атрибуты родителей: ФИО мамы, ФИО папы, дата рождения мамы, дата рождения папы. Атрибуты информации о детях: ФИО, дата рождения, адрес.

Класс будем рассматривать как связь с группой. Атрибуты класса – название, описание, заметки, категория. А Категории рассматриваем как связь с классом. Атрибуты категории – название, описание.

Лист в данной работе рассматривается как связь между такими сущностями как дети, события и группы.

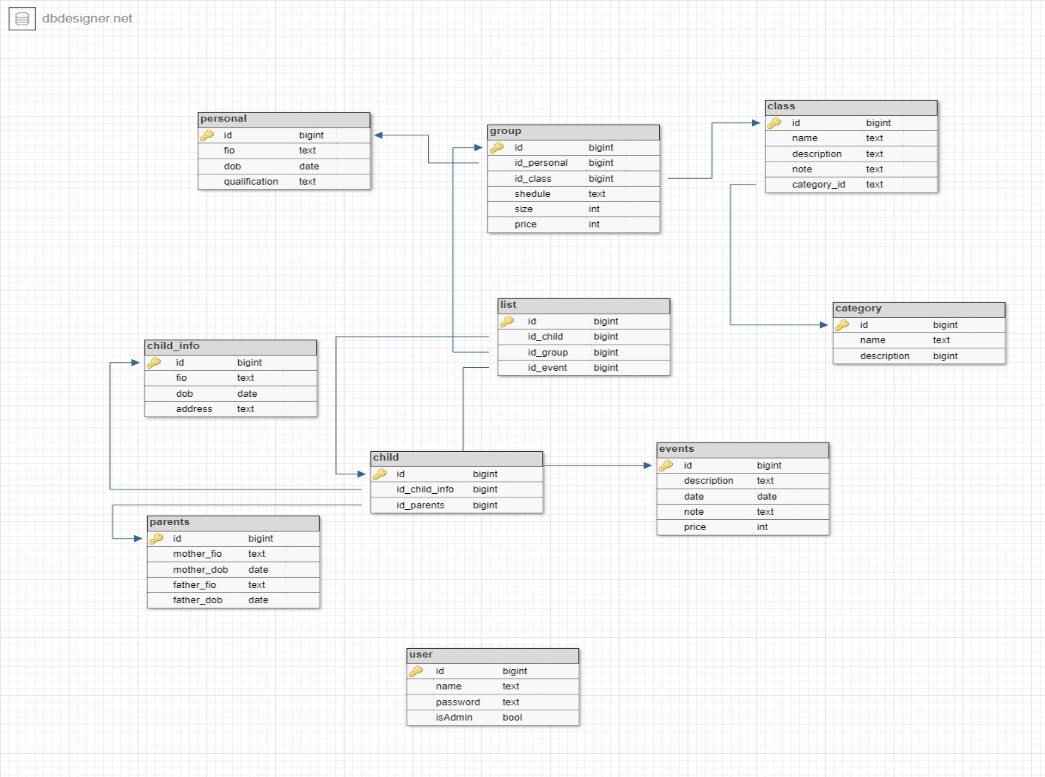


Рисунок 1.1 – ER диаграмма сущностей.

## **1.3 Входные и выходные данные**

Входной информацией в создаваемой программе будут являться запросы пользователей.

Выходной информацией является подробная информация о детях, сотрудниках, группах и событиях представленная в виде таблиц.

# 2. ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

## **2.1 Используемые технические средства**

Для разработки данной программы использовался компьютер со следующими характеристиками:

- Процессор Intel Pentium B960 2.1 Mhz;

- Память ОЗУ 8 GB;

- Дисковый накопитель WDC WD32 00BEVT-80A0RT0 SATA Disk Device (298 Гб);

- ОС Windows 10;

- Видеоадаптер Intel GMA HD.

## **2.2 Инструменты разработки**

Инструментами разработки для будущего программного приложения будут являться:

- операционная система Microsoft Windows 10 Домашняя базовая;

- визуальная среда проектирования Microsoft Visual Studio;

СУБД MySql — предоставляет мощные средства для доступа, настройки, администрирования, разработки всех компонентов базы данных и управления ими. MySql — это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. MySql — свободно распространяемая система. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД. Работать с MySql можно не только в текстовом режиме, но и в графическом. Существует очень популярный визуальный интерфейс для работы с этой СУБД — PhpMyAdmin. Этот интерфейс позволяет значительно упростить работу с базами данных в MySql.

PhpMyAdmin позволяет пользоваться всеми достоинствами браузера, включая прокрутку изображения, если оно не умещается на экран. Многие из базовых SQL-функций работы с данными в PhpMyAdmin сведены к интуитивно понятным интерфейсам и действиям, напоминающим переход по ссылкам в Internet.

Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. VisualStudio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (например, Subversion и VisualSourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода). Главным преимуществом Visual Studio 2017 является производительность. Обеспечивает возможность создания разнообразных приложений на основе одного набора навыков.

**3. ПРОЕКТИРОВАНИЕ ЗАДАЧИ**

**3.1 Концептуальный прототип**

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления. Основной интерфейс программного приложения будут представлять формы.

При проектировании и разработке любой базы данных интерфейс играет важную роль. Он представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными компонентами базы данных.

Работа с базой начинается с авторизации:

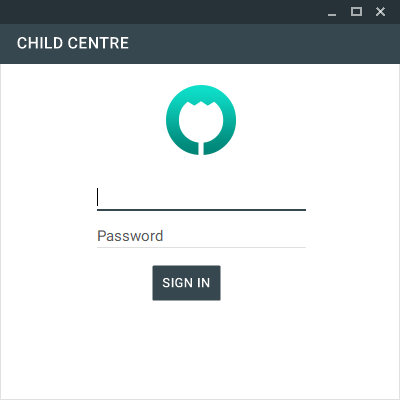


Рисунок 3.1 – Окно авторизации

## При неверно введенных данных появляется окно с ошибкой

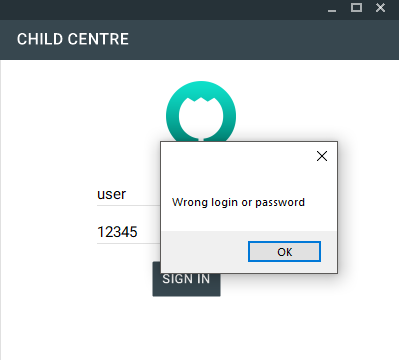
****

Рисунок 3.2 – Ошибка при авторизации

После авторизации в зависимости от роли пользователя можно попасть в форму администратора и форму пользователя, которые предназначены для предоставления пользователю удобной навигации по всем разделами базы данных.

Окно с формами для работы менеджером включает в себя разделы «Группы», «Дети», «События», «Удаленные события», «Удаленные группы» и кнопку выхода из формы. Так же на форме присутствует форма поиска и сортировка данных по таблице.

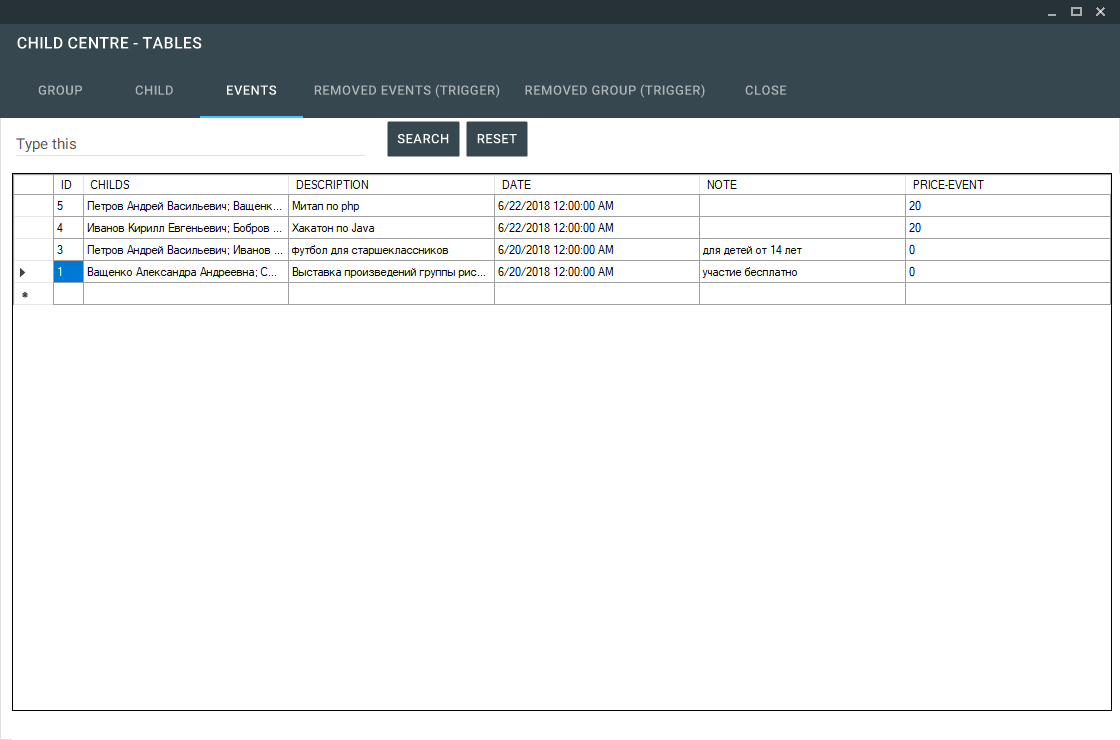


Рисунок 3.3– Окно менеджера

Окно с формами для работы администратора включает в себя разделы «Категории», «Группы», «Дети», «Информация о детях», «События», «Классы», «Список», «Родители», «Персонал» и кнопку выхода из формы. Так же на форме присутствует сортировка данных по таблице.

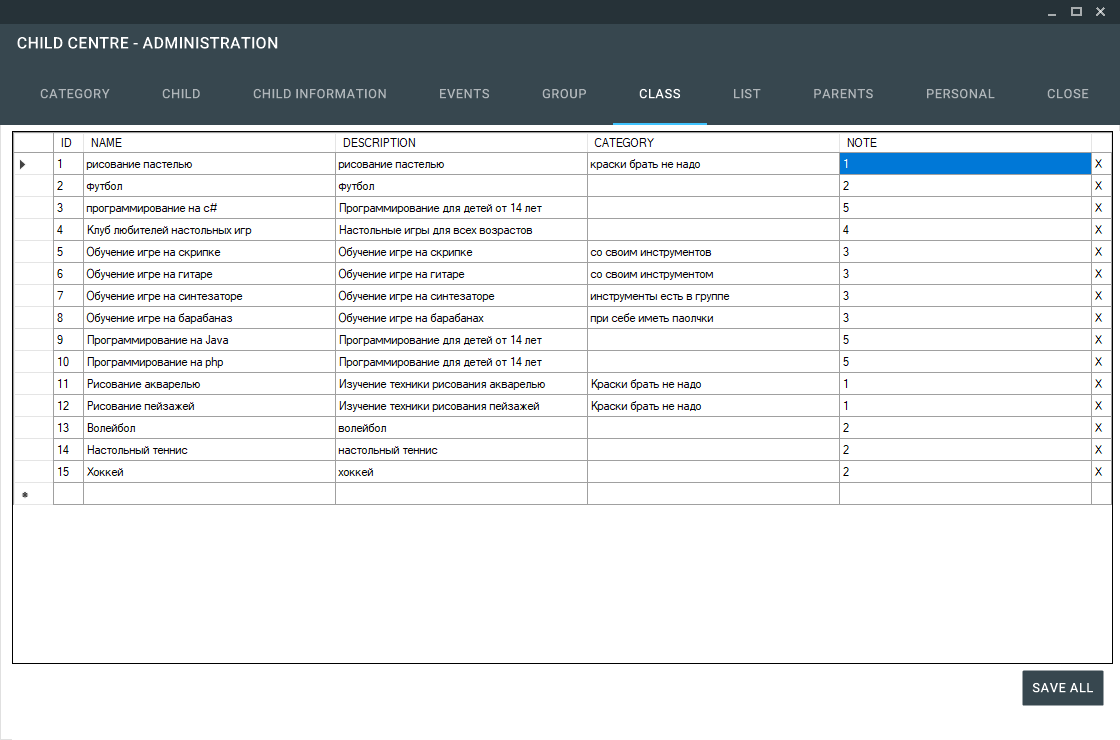


Рисунок 3.4– Окно администратора

Добавление и редактирование пользователей происходит через внесение или редактирование данных в таблице и нажатия кнопки «Sava all». Удаление записей происходит по нажатию на «х» в таблице.

## **3.2 Функции и элементы управления**

В разрабатываемой программе реализуются те задачи и функции, которые рассматривались при постановке задачи.

Программное средство предназначено для функционирования под управлением операционных систем семейства Microsoft Windows, и принципы его функционирования основаны на событийном управлении выполнением приложений, используемых в приложениях Windows. Это означает, что при определенных действиях пользователя, влияющих на состояние элементов управления, система посылает приложению определенные системные события, которые оно может обрабатывать (например, событие, извещающее о нажатии кнопки «Enter»). За определенными элементами управления закрепляются функции, вызываемые при активизации этих элементов управления.

В данном курсовом проекте информационные потребности пользователей реализуются с помощью запросов.

Запрос - это требование на получение определенной информации. Запросы позволяют сфокусировать внимание именно для тех данных, которые нужны для решения текущей задачи.

Используя запросы, можно проверять данные любым образом, который пользователь в состоянии представить. Можно отобрать таблицы, поля и записи, содержащие необходимые для просмотра, подведения итогов или использования в вычислениях данные; отсортировать их; создать отчеты и формы для отображения указанной нами информации и даже создать диаграммы для наглядного представления данных.

Результат работы запроса называется выборкой. Выборка не сохраняется в базе данных; она создается заново каждый раз при выполнении запроса и уничтожается при его закрытии.

Существуют следующие виды запросов:

1. Запрос для отбора данных по заданным сложным условиям из одной или нескольких таблиц баз данных, с группировкой данных для расчета итогов, с показом результатов выполнения запроса в виде таблицы, либо с использованием его для форм и отчетов. После редактирования данных в таблице запроса данные таблиц базы могут обновляться (с некоторыми ограничениями).

2. Перекрестный запрос с формированием двухмерной итоговой таблицы, с группировкой по двум выражениям, одно из которых становится заголовком строки, другое - заголовком столбца.

3. Запрос на создание новой таблицы.

4. Запросы на изменение данных:

* обновление данных - команда занесения общих изменений в группу записей одной или нескольких таблиц;
* добавление данных - команда добавления группы записей из одной или нескольких таблиц в конец одной или нескольких таблиц;
* удаление данных - команда удаления группы записей из одной или нескольких таблиц.

Триггер — это хранимая процедура, которая не вызывается непосредственно, а исполняется при наступлении определенного события (вставка, удаление, обновление строки).

В данном курсовом проекте два триггера. Оба триггера срабатывают при событии удаления в таблицах «Группы» и «События» и добавляют копию удалённой записи в отдельные таблицы «Удалённые группы» и «Удалённые события».

Запрос на создание триггера для таблицы «Группы»:

**CREATE** **DEFINER**=`root`@`%` **TRIGGER** `group\_after\_delete` **AFTER** **DELETE** **ON** `group` **FOR** **EACH** **ROW** **BEGIN**

**INSERT** **INTO** `removed\_group` **Set** id = **OLD**.id, id\_personal = **OLD**.id\_personal, id\_class = **OLD**.id\_class, shedule = **OLD**.shedule, size = **OLD**.size, price = **OLD**.price;

**END**

Запрос на создание триггера для таблицы «События» :

**CREATE** **DEFINER**=`root`@`%` **TRIGGER** `events\_after\_delete` **AFTER** **DELETE** **ON** `events` **FOR** **EACH** **ROW** **BEGIN**

**INSERT** **INTO** `removed\_events` **Set** id = **OLD**.id, description = **OLD**.description, **date** = **OLD**.**date**, note = **OLD**.note, price = **OLD**.price;

**END**

Хранимые процедуры – это способ инкапсуляции повторяющихся действий. В хранимых процедурах можно объявлять переменные, управлять потоками данных, а также применять другие техники программирования.

В данном курсовом проекте используется 2 хранимые процедуры. Первая процедура при её вызове обновляет дату и время авторизации пользователя. Вторая процедура обновляет дату и время выхода из системы пользователя.

Создание хранимой процедуры для обновления даты выхода из системы:

**CREATE** **DEFINER**=`root`@`%` **PROCEDURE** `updateLastLogout`(

**IN** `var1` **INT**

,

**IN** `var1` **INT**

)

**LANGUAGE** **SQL**

**NOT** **DETERMINISTIC**

**CONTAINS** **SQL**

**SQL** **SECURITY** **DEFINER**

**COMMENT** ''

**UPDATE** `user` **SET** `lastLogout`=**CURDATE**() **WHERE** `id`=var1

Создание хранимой процедуры для обновления даты авторизации пользователя:

**CREATE** **DEFINER**=`root`@`%` **PROCEDURE** `updateUserSign`(

**IN** `var1` **INT**

,

**IN** `var1` **INT**

)

**LANGUAGE** **SQL**

**NOT** **DETERMINISTIC**

**CONTAINS** **SQL**

**SQL** **SECURITY** **DEFINER**

**COMMENT** ''

**UPDATE** `user` **SET** `lastSign`=**CURDATE**() **WHERE** `id`=var1

# 4. ОПИСАНИЕ ПРОГРАММНОГО СРЕДСТВА

## **4.1 Общие сведения**

Программный продукт «Система управления «Центр детского творчества» поможет пользователю легко найти нужную информацию о сотруднике, детях, группах и событиях в любом момент времени, автоматизировать все рутинные процессы по учету и управлению данными.

Сфера применения программы «Система управления «Центр детского творчества» это администрирование детских развивающих центров, детских творческих клубов, центров раннего развития и предприятий подобной направленности.

**4.2 Функциональное назначение**

С помощью программы «Центр детского творчества» можно будет выполнять следующие функции:

* Авторизация пользователей;
* Вести клиентскую базу детей с учетом их сопровождающих (родители, родственники);
* Вести клиентскую базу педагогов;
* Проводить мероприятия;
* В удобном виде регистрировать посещения детей на занятия;
* Планировать расписание занятий на любой период;
* Составлять график работы педагогов;
* И много другое…

# 5. МЕТОДИКА ИСПЫТАНИЙ

## **5.1 Технические требования**

Минимальными требованиями для оптимальной работы программного средства, мониторинга нажатия клавиш, является ПК со следующими характеристиками:

- Intel Pentium 4 1600 МГц и выше;

- 512 Мбайт (рекомендуется 1024 Мбайт);

- свободное место на диске 1500 Kb;

- интегрированная видеокарта на 16 Mb и более;

- клавиатура и мышь.

## **5.2 Порядок проведения испытаний**

Тестирование (testing) программного обеспечения (ПО) - это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом. В более широком смысле, тестирование ПО - это техника контроля качества программного продукта, включающая в себя проектирование тестов, выполнение тестирования и анализ полученных результатов.

Очень часто современные программные продукты разрабатываются в сжатые сроки и при ограниченных бюджетах проектов. Программирование сегодня перешло из разряда искусства в разряд ремесел для многих миллионов специалистов. Но, к сожалению, в такой спешке разработчики зачастую игнорируют необходимость обеспечения защищённости своих продуктов, подвергая тем самым пользователей неоправданному риску. Контроль качества (тестирование) считается важным в процессе разработки ПО, потому что обеспечивает безопасность, надёжность, удобство создаваемого продукта. В настоящее время существует великое множество подходов и методик к решению задачи тестирования ПО, но эффективное тестирование сложных программных систем - процесс творческий, не сводящийся к следованию строгим и чётким правилам.

В рамках курсового проекта, программное средство будет протестировано одним из распространенных видов тестирования – функциональным тестированием.

Функциональное тестирование (functional testing) – вид тестирования, направленный на проверку корректности работы функциональности приложения (корректность реализации функциональных требований.

Результаты тестирования программного продукта представлены в таблицах 4.1 – 4.3.

Таблица 4.1 – Результаты тестирования формы авторизации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Условия | Действие | Ожидаемый результат | Итог |
| 1 | 2 | 3 | 4 | 5 |
| 1 | При верном вводе данных для авторизации администратора и нажатии на кнопку «Sign in» | Ввод данных для авторизации и нажатие на кнопку «Sign in» | Авторизация и открытие нового окна с формами для работы от лица администратора | пройден |
| 2 | При верном вводе данных для авторизации пользователя и нажатии на кнопку «Sign in» | Ввод данных для авторизации и нажатие на кнопку «Sign in» | Авторизация и открытие нового окна с формами для работы от лица пользователя | пройден |
| 3 | При неверном вводе данных для авторизации и нажатии на кнопку «Sign in» | Ввод данных для авторизации и нажатие на кнопку «Sign in» | Открытие нового окна с выводом ошибки | пройден |

Таблица 4.2 – Результаты тестирования формы администратора

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Условия | Действие | Ожидаемый результат | Итог |
| 1 | 2 | 3 | 4 | 5 |
| 1 | При нажатии на вкладку | Нажатие на вкладку | Загрузка таблицы с данными, отображение кнопки сохранения данных в таблицу | пройден |
| 2 | При добавлении данных | Ввод данных в таблицу и нажатие кнопки сохранить | Добавление данных в бд и обновление таблицы | пройден |
| 3 | При редактировании данных | Редактирование данных в таблице и нажатие кнопки сохранить | Обновление данных в бд и обновление таблицы | пройден |
| 4 | При удаление данных | Удаление данных в таблице | Удаление данных в бд и обновление таблицы | пройден |

Таблица 4.3 – Результаты тестирования формы пользователя

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Условия | Действие | Ожидаемый результат | Итог |
| 1 | 2 | 3 | 4 | 5 |
| 1 | При нажатии на вкладку | Нажатие на вкладку | Загрузка таблицы с данными | пройден |
| 2 | При поиске | Ввод данных в строку поиска и нажатие на кнопку «Search» | Обновление таблицы в соответствии с условиями поиска | пройден |
| 3 | При сбросе данных поиска | Нажатие на кнопку «Reset» | Обновление таблицы | пройден |
| 4 | При сортировке данных | Нажатие на заголовки таблицы | Сортировка данных в таблице в соответствии с выбранными параметрами | пройден |

Выше были приведены основные проверки форм программного средства. По итоговому результату можно сделать вывод о том, что программное средство выполняет все необходимые функции.

# 6. ПРИМЕНЕНИЕ

## **6.1 Назначение программы**

В процессе разработки было создано программное средство, отвечающее всем требованиям, предъявленным в постановке задачи. Программный продукт «Система управления «Центр детского творчества» поможет пользователю легко найти нужную информацию о сотруднике, детях, группах и событиях в любом момент времени, автоматизировать все рутинные процессы по учету и управлению данными.

## **6.2 Условия применения**

Для использования программного средства не требуется наличие установленной на персональном компьютере среды разработки Microsoft Visual Studio. Программа не требовательна к ресурсам компьютера. Все необходимые зависимости были включены в состав программного обеспечения.

Компьютер, на котором будет установлена данная программа, должен иметь устройство ввода информации, такой как клавиатура, так же, желательно, чтобы был манипулятор «мышь».

**ЗАКЛЮЧЕНИЕ**

Результатом выполнения курсовой работы стало разработанное приложение баз данных, позволяющее автоматизировать рутинные процессы по учету и управлению данными. в детских центрах. Разработанное приложение отвечает всем требованиям предметной области, таблицы созданной базы данных отвечают требованиям нормализации, что позволяет обеспечить целостность и непротиворечивость информации. В данной работе разработана база данных и приложение для работы совместно с базой данных.

В процессе выполнения курсовой работы были закреплены знания, полученные при изучении дисциплины. Были изучены такие пункты:

* анализ предметной области;
* построение концептуальной модели базы данных;
* организация базы данных;
* разработка прикладной программы;
* наполнение и сопровождение базы данных;

В процессе организации БД проведен до необходимого уровня абстракций анализ предметной области, построена реляционная модель БД, произведена нормализация реляционной БД. Реализация проекта была выполнена на современных программных платформах. В качестве технологии доступа к данным была использована объектно-реляционная модель, которая позволяет просто и лаконично осуществлять запросы к базе данных.

Даная курсовая работа была реализована на языке программирования C# с использованием среды разработки Microsoft Visual Studio.

Для проверки корректности работы программы были изучены методы тестирования и проведены тесты, по результатам которых были исправлены ошибки.

Цель и соответствующие задачи, поставленные перед выполнение курсового проекта выполнены в полном объёме.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЯНИКОВ

[1] В.А.Гвоздева, И.Ю.Лаврентьева. Основы построения АИС, Москва, ИД «Форум» - ИНФРА-М, 2009.

[2] А.В.Рудаков, Технология разработки программных продуктов, Москва, Издательский центр «Академия», 2008

[3] Л.Г.Гагарина, Д.В.Киселев, Е.Л.Федотова, Разработка и эксплуатация АИС. Москва, ИД «Форум» - ИНФРА-М, 2009.

[4] Г.Ю.Максимович, А.Г.Романенко, О.Ф.Самойлюк. Информационные системы. Москва 2007, Федеральное агентство по образованию

[5] Диго С.М. Базы данных: проектирование и использование: Учебник. - М.: Финансы и статистика, 2005.

[6] Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных. Учебник для вузов. - М.: Корона-принт, 2004.

[7] Кузин А.В., Левонисова С.В. Базы данных: Учебник. - М.: Academia, 2010.

[8] Джон Дей, Крейг Ван Слайк, Рэймонд Фрост Базы данных. Проектирование и разработка: Учебник. - М.: НТ Пресс, 2007.

[9] Домбровская Г.Р., Новиков Б.А. Настройка приложений баз данных: Учебник - М.: BHV, 2006.

[10] Туманов В.Е. Основы проектирования реляционных баз данных: Учебное пособие. - М.: Интернет-университет информационных технологий, 2010.

[11] Илюшечкин В. М. Основы использования и проектирования баз данных: Учебное пособие. - М.: Юрайт, 2010.

[12] Преснякова Г.В. Проектирование интегрированных реляционных баз данных: Учебник. - М.: КДУ, 2007.

[13] http://www.sql.ru/ - портал про язык SQL и клиент/серверные технологии.

[14] http://sql.itsoft.ru/ - интернет-справочник с примерами по языку SQL.

[15] http://www.cyberguru.ru/database/database-theory/ - статьи по теории баз данных.

ПРИЛОЖЕНИЕ А

Исходный текст программного средства

unit LDSfunc;

interface

uses

Windows,

SysUtils, // Нужен для осуществления файловых операций

Classes, // Также нужен для операций над файлами

ShlObj, ShellAPI, Graphics, Registry,

ActiveX, ComObj, // Нужны для ярлыков

WinSock; // Для определения IP-имени

Const

// Переменные окружения Windows

// нужны для извлечения системных путей

PATH\_DESKTOP = $0000; // Рабочий стол

PATH\_INTERNET = $0001;

PATH\_PROGRAMS = $0002; // "Все программы"

PATH\_CONTROLS = $0003; // Unknowned

PATH\_PRINTERS = $0004;

PATH\_PERSONAL = $0005; // Мои документы

PATH\_FAVORITES = $0006; // Избранное

PATH\_STARTUP = $0007; // Автозагрузка

PATH\_RECENT = $0008; // Ярлыки на недавние документы

PATH\_SENDTO = $0009;

PATH\_BITBUCKET = $000a;

PATH\_STARTMENU = $000b; // Меню "Пуск"

PATH\_DESKTOPDIRECTORY = $0010; // Рабочий стол

PATH\_DRIVES = $0011;

PATH\_NETWORK = $0012;

PATH\_NETHOOD = $0013;

PATH\_FONTS = $0014;

PATH\_TEMPLATES = $0015;

PATH\_COMMON\_STARTMENU = $0016;

PATH\_COMMON\_PROGRAMS = $0017;

PATH\_COMMON\_STARTUP = $0018;

PATH\_COMMON\_DESKTOPDIRECTORY = $0019;

PATH\_APPDATA = $001a;

PATH\_PRINTHOOD = $001b;

PATH\_ALTSTARTUP = $001d;

PATH\_COMMON\_ALTSTARTUP = $001e;

PATH\_COMMON\_FAVORITES = $001f;

PATH\_INTERNET\_CACHE = $0020;

PATH\_COOKIES = $0021;

PATH\_HISTORY = $0022;

PATH\_TEMP = $0023; // My

PATH\_MYMUSIC\_XP = $000d;

PATH\_MYGRAPHICS\_XP = $0027;

PATH\_WINDOWS = $0024; // My

PATH\_SYSTEM = $0025; // My

PATH\_PROGRAMFILES = $0026; // My

PATH\_COMMONFILES = $002b; // My

PATH\_RESOURCES\_XP = $0038;

PATH\_CURRENTUSER\_XP = $0028;

// Константы для извлечения информации о процессоре

CP\_SPEED = $0001; // скорость

CP\_Identifier = $0002; // Стандартное название

CP\_NameString = $0003; // Понятное название

CP\_Usage = $0004; // Загрузка (adCpuUsage.pas)

CP\_VendorIdentifier = $0005; // Производитель

CP\_MMXIdentifier = $0006; // Инф. о сопроцессоре

CP\_Count = $0007; // Кол-во процессоров (adCpuUsage)

// Константы для извлечения информации о ОС

OS\_Version = $0001;

OS\_Platform = $0002;

OS\_Name = $0003;

OS\_Organization = $0004;

OS\_Owner = $0005;

OS\_SerNumber = $0006;

OS\_WinPath = $0007;

OS\_SysPath = $0008;

OS\_TempPath = $0009;

OS\_ProgramFilesPath = $000a;

OS\_IPName = $000b;

// Константы для извлечения информации о памяти

MEM\_TotalPhys = $0001; // Всего физ. памяти

MEM\_AvailPhys = $0002; // Свободно физ. памяти

MEM\_NotAvailPhys = $0003; // Занять физ. памяти

MEM\_TotalPageFile = $0004; // Файл подкачки

MEM\_AvailPageFile = $0005;

MEM\_NotAvailPageFile = $0006;

MEM\_TotalVirtual = $0007; // Виртуальная память

MEM\_AvailVirtual = $0008;

MEM\_NotAvailVirtual = $0009;

// Константы для извлечения информации о диске

DI\_TomLabel = $0001;

DI\_SerNumber = $0002;

DI\_FileSystem = $0003;

DI\_SectorsInClaster = $0004;

DI\_BytesInSector = $0005;

DI\_DiskSize = $0006;

DI\_DiskFreeSize = $0007;

DI\_DiskUsageSize = $0008;

DI\_MaximumLength = $0009; // Максимальная длина пути

DI\_TotalClasters = $000a;

DI\_FreeClasters = $000b;

// Константы для извлечения информации о дисплее

DS\_BitsOnPixel = $0001; // Разрядность цвета

DS\_HorPixelSize = $0002; // Разреш. по горизонтали

DS\_VerPixelSize = $0003; // Разреш. по вертикали

DS\_Frequency = $0004; // Частота мерцаний (XP)

DS\_HorMMSize = $0005; // Ширина в миллиметрах

DS\_VerMMSize = $0006; // Высота в миллиметрах

DS\_PixelsOnInch\_Hor = $0007; // Пикселей на дюйм

DS\_PixelsOnInch\_Ver = $0008; // Пикселей на дюйм

DS\_ColorsInSysPalette = $0009; // Цветов в сис. палитре

// Константы для загрузки параметров окна из ини-файла

WP\_WindowState = $0001;

WP\_Top = $0002;

WP\_Left = $0003;

WP\_Width = $0004;

WP\_Height = $0005;

type

{ Вы можете объявить например следующую функцию:

function DrawCopyProgress(Sour, Dest: String; CopyBytes, Size: DWORD;

Speed: Real; FileNum, Files: Integer): Boolean;

Аргументы следующие:

- Sour - файл-источник,

- Dest - файл-приемник,

- CopyBytes - число скопированных байтов

- Size - размер исходного файла,

- Speed - скорость копирования,

- FileNum - порядковый номер копируемого файла,

- Files - общее число копируемых файлов

Функция должна возвращать значение True. Если она вернет False, то

процесс копирования будет прерван. В дальнейшем функцию DrawCopyProgress

нужно указывать в качестве необязательного параметра к функциям

копирования или перемещения файлов и папок}

TCopyFileProgress = function(Sour, Dest: String; CopyBytes, Size: DWORD;

Speed: Real; FileNum, Files: Integer): Boolean;

{ Работает аналогично TCopyFileProgress, но передает число просмотренных

байт и размер файла. Используется при визуализации поиска в файле}

TFindProgress = function(ByteNum, Bytes: Int64): Boolean;

TLDSfunc = class(TObject)

private

LastSpeed: Real;

function FileSetAttr(const FileName: string; Attr: Integer): Integer;

function FileGetAttr(const FileName: string): Integer;

function \_GetDirList(DirName: String; var List: TStringList): DWORD;

function \_MixLists(var MasterList, DetailList: TStringList): DWORD;

function \_GetFileList(DirName: String; var List: TStringList): DWORD;

function \_GetFilesInfo(DirName: String; var CommonSize: Real): Integer;

function \_DelEmptyPath(DirName, LastDir: String): Boolean;

procedure DoError(Msg: String);

public

(\* Определяет размер файла в байтах \*)

function FileSize(FileName: String): Integer;

(\* Копирует файл с поддержкой визуализации копирования. Необязательные

параметры менять не нужно - они используются другими функциями \*)

function FileCopy(Sour, Dest: String; VisProc: TCopyFileProgress = nil;

FileNum: Integer = 1; Files: Integer = 1): Boolean;

(\* Перемещает файл с поддержкой визуализации копирования. Необязательные

параметры менять не нужно - они используются другими функциями \*)

function FileMove(Sour, Dest: String; VisProc: TCopyFileProgress = nil;

FileNum: Integer = 1; Files: Integer = 1): Boolean;

(\* Осуществляет поиск информации в файле FileName вы можете в качестве

строки поиска задать HEX-последовательность или текст, например:

'aa bb ff'. TypeFind определяет тип поиска. При значении 1 ищется

HEX-последовательность, при 2 ищется текст с учетом регистра, при 3

ищется текст без учета регистра. Возвращает адес начала искомой

информации. Поддерживает визуализацию процесса поиска \*)

function FindHexStringInFile(

FileName: String; // Имя файла

HexString: String; // Строка поиска (HEX-последовательность или текст)

StartByte: DWORD; // Откуда начинать поиск

TypeFind: Byte; // Тип поиска

Func: TFindProgress = nil): DWORD;

(\* Создает полный путь FullPath. Будут созданы все необходимые каталоги \*)

function CreateDir(FullPath: String; DirAttrib: DWORD = 0): Boolean;

(\* Копирование директории. Поддерживается визуализация процесса копирования \*)

function DirCopy(DirSour, DirDest: String; LogFileName: String;

VisProc: TCopyFileProgress = nil): Boolean;

(\* Перемещение директории. Поддерживается визуализация процесса копирования \*)

function DirMove(DirSour, DirDest: String; LogFileName: String;

VisProc: TCopyFileProgress = nil): Boolean;

(\* Удаление директории. Удаляются также все вложенные папки и файлы \*)

function DirDelete(DirName: String; LogFileName: String = ''): Boolean;

(\* Возвращает системный путь \*)

function GetSystemPath(Handle: THandle; PATH\_type: WORD): String;

(\* Возвращает информацию о диске \*)

function GetDiskInfo(DiskName: String; DI\_typeinfo: Byte): String;

(\* Возвращает информацию о состоянии памяти \*)

function GetMemoryInfo(MEM\_typeinfo: Byte): DWORD;

(\* Возвращает информацию об операционной системы \*)

function GetOSInfo(OS\_typeinfo: Byte; Default: String): String;

(\* Определяет реальную частоту процессора \*)

function GetCPUSpeed: Double;

(\* Возвращает информацию о процессоре \*)

function GetProcessorInfo(CP\_typeinfo: Byte;

CP\_NUM: Byte; Default: String): String;

(\* Возвращает информацию о мониторе \*)

function GetDisplaySettings(DS\_typeinfo: Byte): Integer;

(\* Меняет настройки монитора. Указывается нужный режим \*)

function ChangeDisplayMode(ModeNum: Integer): Boolean;

(\* Определяет номер режим дисплея с указанными параметрами. Если

AutoFind=True то можно указать только интересующие параметры,

а остальные задать нулевыми (они не изменятся) \*)

function GetDisplayModeNum(

BitsColor: Byte; // Разрядность пискелей (4,8,16,32)

Dis\_Width, Dis\_Height: Word; // Ширина и высота

Dis\_Freq: Word; // Частота

AutoFind: Boolean

): Integer;

(\* Определяет размер свободного места на диске \*)

function GetFreeDiskSize(Root: String): Real;

(\* Определяет общий размер диска \*)

function GetTotalDiskSize(Root: String): Real;

(\* Универсальная процедура по созданию ярлыка \*)

procedure CreateShotCut(SourceFile, ShortCutName,

SourceParams: String);

(\* Создает ярлык в папке ПРОГРАММЫ меню ПУСК \*)

procedure CreateShotCut\_InPrograms(RelationPath, FullExeName,

SourceParams: String; Handle: THandle);

(\* Создает ярлык на рабочем столе \*)

procedure CreateShotCut\_OnDesktop(FullExeName,

SourceParams: String; Handle: THandle);

(\* Изменяет форму любого визуального компонента, имеющего св-во Handle

в соответствии с указанным bmp-файлом \*)

function CreateRgnFromBitmap(Handle: THandle; BmpFileName: TFileName): HRGN;

(\* Запуск файла на выполнение \*)

function ExecuteFile

(WND: HWND;

const FileName, // Имя запускаемого файла (с возможным путем доступа)

Params, // Передаваемые параметры

DefaultDir: string;// Текущая директория для запускаемой программы

ShowCmd: Integer // Способ отображения окна запущенного файла

): THandle;

(\* Определение IP-имени данного компьютера \*)

function GetIPName: String;

(\* Генерация ХЭШ-кода текста по алгоритму МД-5 \*)

function md5(s: string):string;

(\* Правка HEX-строки \*)

function StringToHex(HexStr: String): String;

(\* Исключение замыкающего Backslash (чтобы Делфи не ругался) \*)

function ExcludeTrailingBackslash(const S: string): string;

end;

var

LDSf: TLDSfunc;

implementation

{ TLDSfunc }

function TLDSfunc.CreateDir(FullPath: String; DirAttrib: DWORD = 0): Boolean;

var

List: TStringList;

I, J, K, C: Integer;

begin

K := 0;

C := 0;

Result := False;

if not DirectoryExists(FullPath[1] + ':\') then Exit;

if Length(FullPath)<4 then Exit;

if not(((FullPath[1]>='A') and (FullPath[1]<='Z')) or

((FullPath[1]>='a') and (FullPath[1]<='z'))) then Exit;

if (FullPath[2]<>':') or (FullPath[3]<>'\') then Exit;

FullPath := ExcludeTrailingBackslash(FullPath) + '\';

for I := 5 to Length(FullPath) do

if FullPath[I] = '\' then Inc(C);

// Теперь С хранить возможное количество путей для создания

// Создаем список путей

List :=TStringList.Create;

for I:=0 to C-1 do // Перебор возможных путей

begin

List.Append('');

for J:=1 to Length(FullPath) do // Формируем путь

begin

if (FullPath[J] = '\') and (J>4) and (J>K) then

begin

K := J;

Break;

end;

List.Strings[I] := List.Strings[I]+ FullPath[J];

end;

end; // Все пути сформированы

//Теперь в цикле создаем директории

for I := 0 to C-1 do

if not DirectoryExists(List.Strings[I]) then

try

MkDir(List.Strings[I]);

except

MessageBox(GetActiveWindow,PChar('Невозможно создать директорию "'+

List.Strings[I]+'"'+#13#10+

'Проверьте наличие устройства "'+AnsiUpperCase(FullPath[1])+':\"'),

'Ошибка ввода/вывода',MB\_OK or MB\_ICONSTOP or MB\_TOPMOST);

Exit;

end;

List.Free; // Освобождаем память из-под объекта

// Устанавливаем атрибуты конечной папки

SetFileAttributes(PChar(FullPath), DirAttrib);

Result := True;

end;

function TLDSfunc.DirCopy(DirSour, DirDest: String; LogFileName: String;

VisProc: TCopyFileProgress = nil): Boolean;

var

DirList, //Хранит список всех возможных каталогов

TempList: TStringList;

C, K: DWORD;

S: String;

I, Files, FileNum: Integer;

tf: TextFile;

tk: DWORD;

CommSize, Tmp: Real;

begin

Result := False;

if not DirectoryExists(DirSour) then DoError('Директория-источник не найдена');

if not DirectoryExists(DirDest[1]+':\') then DoError('Накопитель-приемник не найдена');

tk := GetTickCount;

// Записываем в ЛОГ-файл информацию о начале сессии

if LogFileName <> '' then

begin

if not FileExists(LogFileName) then

begin

AssignFile(tf, LogFileName);

ReWrite(tf);

CloseFile(tf);

end;

try

AssignFile(tf, LogFileName);

Append(tf);

Writeln(tf,'' );

Writeln(tf,'' );

Writeln(tf,'\*\*\*\* НАЧАЛО СЕССИИ КОПИРОВАНИЯ: '+DateTimeToStr(Now)+' \*\*\*\*');

Writeln(tf,'' );

except

DoError('Ошибка при создании файла отчета');

end;

end;

DirList := TStringList.Create;

TempList := TStringList.Create;

if DirSour[Length(DirSour)] <> '\' then DirSour := DirSour+'\';

if DirDest[Length(DirDest)] <> '\' then DirDest := DirDest+'\';

// Создаем корневой каталог-приемник

if not CreateDir(DirDest,0) then begin

if LogFileName <> '' then CloseFile(tf);

DoError('Не смог создать приемную директорию');

end;

C:= \_GetDirList(DirSour,DirList); // Получаем первый список каталогов

I:=0;

If C > 0 then

While true do

begin

TempList.Clear;

\_GetDirList(DirList.Strings[I],TempList);

\_MixLists(DirList,TempList);

If I = DirList.Count-1 then Break;

Inc(I);

end; // While

DirList.Sort;

FileNum := 0;

Files := 0;

CommSize := 0;

if DirList.Count > 0 then

for I := 0 to DirList.Count-1 do begin

Files := Files + \_GetFilesInfo(DirList[I], Tmp);

CommSize := CommSize + Tmp;

end;

Files := Files + \_GetFilesInfo(DirSour, Tmp);

CommSize := CommSize + Tmp;

if CommSize > GetFreeDiskSize(DirDest[1]) then begin

if LogFileName <> '' then CloseFile(tf);

DoError('Не хватает дисковой памяти для копирования директории');

end;

// Создаем каталоги из списка на получателе

if DirList.Count>0 then

for I := 0 to DirList.Count-1 do

begin

S := Copy(DirList.Strings[I], Length(DirSour)+1,

Length(DirList.Strings[I])-Length(DirSour)+1);

// Cоздает нужный каталог

if not

CreateDir(DirDest+S,GetFileAttributes(PChar(DirList.Strings[I])))

then begin

if LogFileName <> '' then CloseFile(tf);

DoError('Не смог создать приемную директорию');

end;

end;

// Копируем файлы

if DirList.Count>0 then

for I := 0 to DirList.Count-1 do

begin

TempList.Clear;

C:= \_GetFileList(DirList.Strings[I],TempList);

if C>0 then

for K:=0 to C-1 do

begin

S := Copy(DirList.Strings[I], Length(DirSour)+1,

Length(DirList.Strings[I])-Length(DirSour)+1);

if LogFileName <> '' then

Write(tf,TimeToStr(Time)+' --Copy-- '+

DirList.Strings[I]+'\'+TempList.Strings[K]+' --> '+

DirDest+S+'\'+TempList.Strings[K]);

Inc(FileNum);

try

if not FileCopy(DirList.Strings[I]+'\'+TempList.Strings[K],

DirDest+S+'\'+TempList.Strings[K], VisProc, FileNum, Files) then begin

if LogFileName <> '' then CloseFile(tf);

Exit;

end;

except

if MessageBox(GetActiveWindow,'Произошла ошибка при работе с файлами'+

#13#10+'Продолжить со следующего файла?', 'Предупреждение',

MB\_YESNO or MB\_ICONEXCLAMATION) = IDNO then begin

if LogFileName <> '' then CloseFile(tf);

Exit;

end;

end;

if LogFileName <> '' then

WriteLn(tf,' --OK-- ');

end;

end;

// Теперь делаем, чтобы копировались файлы из самой выбранной папки

C:= \_GetFileList(DirSour,TempList);

if C > 0 then

for I := 0 to C-1 do

begin

S := DirDest+TempList.Strings[I]; // Куда копируем

if LogFileName <> '' then

Write(tf,TimeToStr(Time)+' --Copy-- '+

DirSour+TempList.Strings[I]+' --> '+

S);

Inc(FileNum);

try

if not FileCopy(DirSour+TempList.Strings[I],S, VisProc, FileNum, Files) then begin

if LogFileName <> '' then CloseFile(tf);

Exit;

end;

except

if MessageBox(GetActiveWindow,'Произошла ошибка при работе с файлами'+

#13#10+'Продолжить со следующего файла?', 'Предупреждение',

MB\_YESNO or MB\_ICONEXCLAMATION) = IDNO then begin

if LogFileName <> '' then CloseFile(tf);

Exit;

end;

end;

if LogFileName <> '' then

WriteLn(tf,' --OK-- ');

end;

DirList.Free;

TempList.Free;

if LogFileName <> '' then

begin

WriteLn(tf,'');

WriteLn(tf,'\*\*\*\* Время сессии: '+TimeToStr((Gettickcount-tk)\*

StrToTime('00:00:01')/1000)+ ' \*\*\*\*');

WriteLn(tf,'');

CloseFile(tf);

end;

Result := True;

end;

procedure TLDSfunc.DoError(Msg: String);

begin

raise Exception.Create(Msg);

end;

function TLDSfunc.ExcludeTrailingBackslash(const S: string): string;

begin

Result := ExcludeTrailingPathDelimiter(S);

end;

function TLDSfunc.FileCopy(Sour, Dest: String; VisProc: TCopyFileProgress = nil;

FileNum: Integer = 1; Files: Integer = 1): Boolean;

var

ArrayRead: Array[1..8192] Of Byte;

FP\_Read,FP\_Write : TFileStream;

fSize, I: Integer;

sfHandle: HFile;

fTime, CurIter: Integer;

PartTime, Speed: Real;

Tc, Delta: Cardinal;

begin

Result := False;

if not DirectoryExists(ExtractFileDir(Dest)) then DoError('Директории не существует');

if not FileExists(Sour) then DoError('Файл не найден');

if Dest = Sour then begin Result := True; Exit; end;

if FileExists(Dest) then

begin

FileSetAttr(Dest, 0); // Сбрасываем все атрибуты, чтобы можно было удалить

DeleteFile(Dest);

end;

try

FP\_Read := TFileStream.Create(Sour,fmOpenRead);

fSize := FP\_Read.Size;

FP\_Write := TFileStream.Create(Dest,fmCreate);

except

DoError('Невозможно получить доступ к файлу.'+

' Возможно он используется другим приложением'); Exit;

end;

CurIter := 0;

Tc := GetTickCount;

while FP\_Read.Position < FSize do

begin

Inc(CurIter);

// код для обработки сообщений закончен

I := FSize-FP\_Read.Position;

if I >= 8192 then

begin

FP\_Read.Read(ArrayRead,8192);

FP\_Write.Write(ArrayRead,8192);

end else

begin

FP\_Read.Read(ArrayRead,I);

FP\_Write.Write(ArrayRead,I);

end;

// Через каждые пол секунды выполняем визуализацию

if (@VisProc <> nil) and ((GetTickCount - Tc > 500) or

(FP\_Read.Position = fSize)) then

begin

Delta := GetTickCount - Tc;

PartTime := Delta / 1000;

if Delta < 50 then Speed := LastSpeed else

Speed := CurIter \* 8192 / PartTime;

LastSpeed := Speed;

Tc := GetTickCount;

CurIter := 0;

if VisProc(Sour, Dest, FP\_Read.Position, fSize, Speed, FileNum,

Files) = False then begin

FP\_Read.Free;

FP\_Write.Free;

if FileExists(Dest) then DeleteFile(Dest);

Exit;

end;

end;

end;

FP\_Read.Free;

FP\_Write.Free;

// Устанавливает атрибуты

sfHandle:=\_lopen(PAnsiChar(Sour),OF\_READ);

fTime:=FileGetDate(sfHandle);

\_lclose(sfHandle);

FileSetDate(Dest,fTime);

FileSetAttr(Dest, FileGetAttr(Sour));

Result := True;

end;

function TLDSfunc.FileGetAttr(const FileName: string): Integer;

begin

Result := GetFileAttributes(PChar(FileName));

end;

function TLDSfunc.FileMove(Sour, Dest: String; VisProc: TCopyFileProgress = nil;

FileNum: Integer = 1; Files: Integer = 1): Boolean;

var

FullDest: String;

begin

Result := True;

FullDest:= ExpandFileName(Dest);

if Sour = FullDest then Exit;

if not DirectoryExists(FullDest[1]+':\') then DoError('Имя приемника задано неверно');

if not FileExists(Sour) then DoError('Исходный файл не был найден');

// Если диск один, то можно переименовать старый файл

if not RenameFile(Sour, FullDest) then

begin

FileCopy(Sour, FullDest, VisProc, FileNum, Files);

FileSetAttr(Sour,0);

DeleteFile(Sour);

end;

end;

function TLDSfunc.FileSetAttr(const FileName: string;

Attr: Integer): Integer;

begin

Result := 0;

if not SetFileAttributes(PChar(FileName), Attr) then

Result := GetLastError;

end;

function TLDSfunc.FileSize(FileName: String): Integer;

var

F: HFile; // Для расчета размера

begin

Result := 0;

if not FileExists(FileName) then Exit;

F:=\_lopen(PAnsiChar(FileName), OF\_READ);

Result :=\_llseek(F,0, FILE\_END);

\_lclose(F);

end;

function TLDSfunc.\_GetDirList(DirName: String;

var List: TStringList): DWORD;

var

FS: TSearchRec;

Path: String;

begin

if DirName[Length(DirName)]<>'\' then DirName := DirName+'\';

Path := DirName+'\*.\*';

List.Clear;

List.Sort;

if FindFirst(Path, faAnyFile, FS)=0 then

begin

if (DirectoryExists(DirName+FS.Name))

and ((FS.Name<>'.') and (FS.Name<>'..')) then

List.Append(DirName+FS.Name);

while FindNext(FS)=0 do

begin

if (DirectoryExists(DirName+FS.Name))

and ((FS.Name<>'.') and (FS.Name<>'..'))

then

List.Append(DirName+FS.Name);

end;

FindClose(FS);

end;

Result := List.Count;

end;

function TLDSfunc.\_GetFilesInfo(DirName: String; var CommonSize: Real): Integer;

var

FS: TSearchRec;

Path: String;

begin

if DirName[Length(DirName)]<>'\' then DirName := DirName+'\';

Path := DirName+'\*.\*';

Result := 0;

CommonSize := 0;

if FindFirst(Path, $00000020, FS)=0 then

begin

Inc(Result);

CommonSize := CommonSize + FS.Size;

while FindNext(FS)=0 do begin

CommonSize := CommonSize + FS.Size;

Inc(Result);

end;

FindClose(FS);

end;

end;

function TLDSfunc.\_GetFileList(DirName: String;

var List: TStringList): DWORD;

var

FS: TSearchRec;

Path: String;

begin

if DirName[Length(DirName)]<>'\' then DirName := DirName+'\';

Path := DirName+'\*.\*';

List.Clear;

if FindFirst(Path, $00000020, FS)=0 then

begin

List.Append(FS.Name);

while FindNext(FS)=0 do

begin

List.Append(FS.Name);

end;

FindClose(FS);

end;

Result := List.Count;

end;

function TLDSfunc.\_MixLists(var MasterList,

DetailList: TStringList): DWORD;

var D, C: DWORD;

begin

C :=DetailList.Count;

if C<>0 then

for D := 0 to C-1 do

MasterList.Append(DetailList.Strings[D]);

Result := MasterList.Count;

end;

function TLDSfunc.GetDiskInfo(DiskName: String; DI\_typeinfo: Byte): String;

var

lpRootPathName: PChar;

lpVolumeNameBuffer: PChar;

nVolumeNameSize: DWORD;

lpVolumeSerialNumber: DWORD;

lpMaximumComponentLength: DWORD;

lpFileSystemFlags: DWORD;

lpFileSystemNameBuffer: PChar;

nFileSystemNameSize: DWORD;

FSectorsPerCluster: DWORD;

FBytesPerSector : DWORD;

FFreeClusters : DWORD;

FTotalClusters : DWORD;

begin

DiskName := DiskName[1];

if not DirectoryExists(DiskName+':\') then

begin

Result := '0';

Exit;

end;

lpVolumeSerialNumber := 0;

lpMaximumComponentLength:= 0;

lpFileSystemFlags := 0;

GetMem(lpVolumeNameBuffer, MAX\_PATH + 1);

GetMem(lpFileSystemNameBuffer, MAX\_PATH + 1);

nVolumeNameSize := MAX\_PATH + 1;

nFileSystemNameSize := MAX\_PATH + 1;

lpRootPathName := PChar(DiskName+':\');

if GetVolumeInformation( lpRootPathName, lpVolumeNameBuffer,

nVolumeNameSize, @lpVolumeSerialNumber, lpMaximumComponentLength,

lpFileSystemFlags, lpFileSystemNameBuffer, nFileSystemNameSize ) then

begin

GetDiskFreeSpace( PChar(DiskName+':\'),

FSectorsPerCluster, FBytesPerSector, FFreeClusters, FTotalClusters);

case DI\_typeinfo of

DI\_SerNumber: Result:=IntToHex(HIWord(lpVolumeSerialNumber), 4) + '-' +

IntToHex(LOWord(lpVolumeSerialNumber), 4);

DI\_TomLabel: Result := lpVolumeNameBuffer;

DI\_FileSystem:Result := lpFileSystemNameBuffer;

DI\_SectorsInClaster:Result := IntToStr(FSectorsPerCluster);

DI\_BytesInSector: Result := IntToStr(FBytesPerSector);

DI\_MaximumLength: Result := IntToStr(lpMaximumComponentLength);

DI\_DiskFreeSize: Result := FloatToStr(GetFreeDiskSize(DiskName));

DI\_DiskSize: Result := FloatToStr(GetTotalDiskSize(DiskName));

DI\_DiskUsageSize: Result := FloatToStr(GetTotalDiskSize(DiskName)-

GetFreeDiskSize(DiskName));

DI\_TotalClasters: Result := IntToStr(FTotalClusters);

DI\_FreeClasters: Result := IntToStr(FFreeClusters);

end; // Case

end; // If

FreeMem(lpVolumeNameBuffer);

FreeMem(lpFileSystemNameBuffer);

end;

function TLDSfunc.GetFreeDiskSize(Root: String): Real;

begin

Result := 0;

if not DirectoryExists(Root[1]+':\') then Exit;

Root := UpperCase(Root);

Result := DiskFree(Ord(Root[1]) - 64 );

end;

function TLDSfunc.GetTotalDiskSize(Root: String): Real;

begin

Result := 0;

if not DirectoryExists(Root[1]+':\') then Exit;

Root := UpperCase(Root);

Result := DiskSize(Ord(Root[1]) - 64 );

end;

function TLDSfunc.DirMove(DirSour, DirDest, LogFileName: String;

VisProc: TCopyFileProgress): Boolean;

begin

Result := False;

if not DirectoryExists(DirSour) then Exit;

if not DirectoryExists(DirDest[1]+':\') then Exit;

if DirSour = DirDest then

begin

Result := True;

Exit;

end;

if not CreateDir(DirDest,0) then Exit; // Создаем полный путь

if not RemoveDir(DirDest) then Exit; // Удаляем последний каталог

// Если указан существующий каталог, то функция пытается переместить файлы

// в него, однако если в нем уже есть к.-л. файлы, то работа функции

// прерывается (возвращается False)

// Если источник и приемник находятся на одном диске,

// то попытаемся переименовать источник в приемник

if not RenameFile(DirSour, DirDest) then

begin

DirCopy(DirSour, DirDest,LogFileName, VisProc);

DirDelete(DirSour, LogFileName);

end;

Result:=True;

end;

function TLDSfunc.DirDelete(DirName: String; LogFileName: String = ''): Boolean;

var

tf: TextFile;

tk: DWORD;

DirList, //Хранит список всех возможных каталогов

TempList: TStringList;

C, K: DWORD;

I: Integer;

begin

Result := True;

if not DirectoryExists(DirName) then Exit;

tk := GetTickCount;

// Записываем в ЛОГ-файл информацию о начале сессии

if LogFileName <> '' then

begin

if not FileExists(LogFileName) then

begin

AssignFile(tf, LogFileName);

ReWrite(tf);

CloseFile(tf);

end;

AssignFile(tf, LogFileName);

Append(tf);

Writeln(tf,'' );

Writeln(tf,'' );

Writeln(tf,'\*\*\*\* НАЧАЛО СЕССИИ УДАЛЕНИЯ: '+DateTimeToStr(Now)+' \*\*\*\*');

Writeln(tf,'' );

end; // if

DirList := TStringList.Create;

TempList := TStringList.Create;

if DirName[Length(DirName)] <> '\' then DirName := DirName+'\';

C:= \_GetDirList(DirName,DirList); // Получаем первый список каталогов

I:=0;

If C > 0 then

While true do

begin

TempList.Clear;

\_GetDirList(DirList.Strings[I],TempList);

\_MixLists(DirList,TempList);

If I=DirList.Count-1 then Break;

Inc(I);

end; // While

// Теперь нам известны все пути для удаления

DirList.Sort;

DirList.Append(ExcludeTrailingBackslash(DirName)); // Добавляем корневой каталог в конец списка

// Начинаем удаление файлов из всех каталогов

if DirList.Count>0 then

for I := 0 to DirList.Count - 1 do

begin

// Получаем список файлов для текущего каталога

C := \_GetFileList(DirList.Strings[I], TempList);

if C > 0 then

for K := 0 to C-1 do

begin

if LogFileName <> '' then

Write(tf,TimeToStr(Time)+' --Delete-- '+

DirList.Strings[I]+'\'+TempList.Strings[K]);

SetFileAttributes(PChar(DirList.Strings[I]+'\'+TempList.Strings[K]),0);

if DeleteFile(PChar(DirList.Strings[I]+'\'+TempList.Strings[K])) then

if LogFileName <> '' then

WriteLn(tf,' --OK--') else WriteLn(tf,' --NO--');

end;

end;

// Теперь удаляем сами каталоги

if DirList.Count >0 then

for I := 0 to DirList.Count-1 do

begin

// Здесь возникает проблема из-за того, что удалять каталоги

// нужно по порядку - от дочернего к родительскому

\_DelEmptyPath(DirList.Strings[I], DirName) ;

end;

if LogFileName <> '' then

begin

WriteLn(tf,'');

WriteLn(tf,'\*\*\*\* Время сессии: '+TimeToStr((Gettickcount-tk)\*

StrToTime('00:00:01')/1000)+ ' \*\*\*\*');

WriteLn(tf,'');

CloseFile(tf);

end;

if DirectoryExists(DirName) then Result := False else Result := True;

DirList.Free;

TempList.Free;

end;

function TLDSfunc.\_DelEmptyPath(DirName, LastDir: String): Boolean;

var

DirList, TempList: TStringList;

I: DWORD;

begin

Result := False;

if not DirectoryExists(DirName) then Exit;

DirName:=ExcludeTrailingBackslash(DirName)+'\';

LastDir:=ExcludeTrailingBackslash(LastDir)+'\';

DirList:= TStringList.Create;

TempList:= TStringList.Create;

if Length(DirName)<5 then Exit;

if Length(LastDir)>Length(DirName) then Exit;

for I := Length(DirName) downto Length(LastDir)+1 do

begin

if DirName[I]='\' then

DirList.Append(Copy(DirName,1,I-1));

end;

// Добавляем последнюю директорию

DirList.Append(ExcludeTrailingBackslash(LastDir));

if DirList.Count > 0 then

for I := 0 to DirList.Count - 1 do

if \_GetDirList(DirList.Strings[I], TempList) = 0 then

begin

FileSetAttr(DirList.Strings[I],0);

RemoveDir(DirList.Strings[I]);

end;

Result := True;

DirList.Free;

TempList.Free;

end;

function TLDSfunc.GetMemoryInfo(MEM\_typeinfo: Byte): DWORD;

var

MemInfo : TMemoryStatus;

begin

Result := 0;

MemInfo.dwLength := Sizeof (MemInfo);

GlobalMemoryStatus (MemInfo);

Case MEM\_typeinfo of

MEM\_TotalPhys: Result := MemInfo.dwTotalPhys;

MEM\_AvailPhys : Result := MemInfo.dwAvailPhys;

MEM\_NotAvailPhys : Result := MemInfo.dwTotalPhys-MemInfo.dwAvailPhys;

MEM\_TotalPageFile: Result := MemInfo.dwTotalPageFile;

MEM\_AvailPageFile : Result := MemInfo.dwAvailPageFile;

MEM\_NotAvailPageFile : Result := MemInfo.dwTotalPageFile-MemInfo.dwAvailPageFile;

MEM\_TotalVirtual: Result := MemInfo.dwTotalVirtual;

MEM\_AvailVirtual: Result := MemInfo.dwAvailVirtual;

MEM\_NotAvailVirtual : Result := MemInfo.dwTotalVirtual-MemInfo.dwAvailVirtual;

end;

end;

function TLDSfunc.GetDisplaySettings(DS\_typeinfo: Byte): Integer;

var

DC: HDC;

begin

Result := 0;

DC := GetDC(0); // Канва Десктопа всегда 0

Case DS\_typeinfo of

DS\_BitsOnPixel : Result := GetDeviceCaps(DC, BITSPIXEL);

DS\_HorPixelSize : Result := GetDeviceCaps(DC, HORZRES);

DS\_VerPixelSize : Result := GetDeviceCaps(DC, VERTRES);

DS\_Frequency : Result := GetDeviceCaps(DC, VREFRESH);

DS\_HorMMSize : Result := GetDeviceCaps(DC, HORZSIZE);

DS\_VerMMSize : Result := GetDeviceCaps(DC, VERTSIZE);

DS\_PixelsOnInch\_Hor:Result := GetDeviceCaps(DC, LOGPIXELSX);

DS\_PixelsOnInch\_Ver:Result := GetDeviceCaps(DC, LOGPIXELSY);

DS\_ColorsInSysPalette:Result:= GetDeviceCaps(DC, SIZEPALETTE);

end;

end;

function TLDSfunc.ChangeDisplayMode(ModeNum: Integer): Boolean;

var

Mode: TDevMode;

begin

Result := False;

if EnumDisplaySettings(nil, ModeNum, Mode) then begin

Mode.dmFields := DM\_BITSPERPEL or

DM\_PELSWIDTH or DM\_PELSHEIGHT or DM\_DISPLAYFLAGS

or DM\_DISPLAYFREQUENCY;

ChangeDisplaySettings(Mode, CDS\_UPDATEREGISTRY);

Result := True;

end;

end;

function TLDSfunc.GetDisplayModeNum(BitsColor: Byte; Dis\_Width, Dis\_Height,

Dis\_Freq: Word; AutoFind: Boolean): Integer;

var

modes: array[0..255] of TDevMode;

I: Integer;

begin

Result := -1;

if not AutoFind then

if (BitsColor=0)or (Dis\_Width=0)or(Dis\_Height=0) or (Dis\_Freq=0) then Exit;

if AutoFind then

begin

if BitsColor = 0 then BitsColor:=GetDisplaySettings(DS\_BitsOnPixel);

if Dis\_Width = 0 then Dis\_Width:=GetDisplaySettings(DS\_HorPixelSize);

if Dis\_Height = 0 then Dis\_Height:=GetDisplaySettings(DS\_VerPixelSize);

end;

I:=0;

while EnumDisplaySettings(nil, I, Modes[I]) do

begin

if (BitsColor=modes[I].dmBitsPerPel) AND

(Dis\_Width=modes[I].dmPelsWidth) AND

(Dis\_Height=modes[I].dmPelsHeight) then

if (AutoFind AND (Dis\_Freq = 0)) or (Dis\_Freq=modes[I].dmDisplayFrequency)

then Result := I;

Inc(I);

end;

end;

function TLDSfunc.md5(s: string): string;

var

a:array[0..15] of byte;

i:integer;

LenHi, LenLo: longword;

Index: DWord;

HashBuffer: array[0..63] of byte;

CurrentHash: array[0..3] of DWord;

procedure Burn;

begin

LenHi:= 0; LenLo:= 0;

Index:= 0;

FillChar(HashBuffer,Sizeof(HashBuffer),0);

FillChar(CurrentHash,Sizeof(CurrentHash),0);

end;

procedure Init;

begin

Burn;

CurrentHash[0]:= $67452301;

CurrentHash[1]:= $efcdab89;

CurrentHash[2]:= $98badcfe;

CurrentHash[3]:= $10325476;

end;

function LRot32(a, b: longword): longword;

begin

Result:= (a shl b) or (a shr (32-b));

end;

procedure Compress;

var

Data: array[0..15] of dword;

A, B, C, D: dword;

begin

Move(HashBuffer,Data,Sizeof(Data));

A:= CurrentHash[0];

B:= CurrentHash[1];

C:= CurrentHash[2];

D:= CurrentHash[3];

A:= B + LRot32(A + (D xor (B and (C xor D))) + Data[ 0] + $d76aa478,7);

D:= A + LRot32(D + (C xor (A and (B xor C))) + Data[ 1] + $e8c7b756,12);

C:= D + LRot32(C + (B xor (D and (A xor B))) + Data[ 2] + $242070db,17);

B:= C + LRot32(B + (A xor (C and (D xor A))) + Data[ 3] + $c1bdceee,22);

A:= B + LRot32(A + (D xor (B and (C xor D))) + Data[ 4] + $f57c0faf,7);

D:= A + LRot32(D + (C xor (A and (B xor C))) + Data[ 5] + $4787c62a,12);

C:= D + LRot32(C + (B xor (D and (A xor B))) + Data[ 6] + $a8304613,17);

B:= C + LRot32(B + (A xor (C and (D xor A))) + Data[ 7] + $fd469501,22);

A:= B + LRot32(A + (D xor (B and (C xor D))) + Data[ 8] + $698098d8,7);

D:= A + LRot32(D + (C xor (A and (B xor C))) + Data[ 9] + $8b44f7af,12);

C:= D + LRot32(C + (B xor (D and (A xor B))) + Data[10] + $ffff5bb1,17);

B:= C + LRot32(B + (A xor (C and (D xor A))) + Data[11] + $895cd7be,22);

A:= B + LRot32(A + (D xor (B and (C xor D))) + Data[12] + $6b901122,7);

D:= A + LRot32(D + (C xor (A and (B xor C))) + Data[13] + $fd987193,12);

C:= D + LRot32(C + (B xor (D and (A xor B))) + Data[14] + $a679438e,17);

B:= C + LRot32(B + (A xor (C and (D xor A))) + Data[15] + $49b40821,22);

A:= B + LRot32(A + (C xor (D and (B xor C))) + Data[ 1] + $f61e2562,5);

D:= A + LRot32(D + (B xor (C and (A xor B))) + Data[ 6] + $c040b340,9);

C:= D + LRot32(C + (A xor (B and (D xor A))) + Data[11] + $265e5a51,14);

B:= C + LRot32(B + (D xor (A and (C xor D))) + Data[ 0] + $e9b6c7aa,20);

A:= B + LRot32(A + (C xor (D and (B xor C))) + Data[ 5] + $d62f105d,5);

D:= A + LRot32(D + (B xor (C and (A xor B))) + Data[10] + $02441453,9);

C:= D + LRot32(C + (A xor (B and (D xor A))) + Data[15] + $d8a1e681,14);

B:= C + LRot32(B + (D xor (A and (C xor D))) + Data[ 4] + $e7d3fbc8,20);

A:= B + LRot32(A + (C xor (D and (B xor C))) + Data[ 9] + $21e1cde6,5);

D:= A + LRot32(D + (B xor (C and (A xor B))) + Data[14] + $c33707d6,9);

C:= D + LRot32(C + (A xor (B and (D xor A))) + Data[ 3] + $f4d50d87,14);

B:= C + LRot32(B + (D xor (A and (C xor D))) + Data[ 8] + $455a14ed,20);

A:= B + LRot32(A + (C xor (D and (B xor C))) + Data[13] + $a9e3e905,5);

D:= A + LRot32(D + (B xor (C and (A xor B))) + Data[ 2] + $fcefa3f8,9);

C:= D + LRot32(C + (A xor (B and (D xor A))) + Data[ 7] + $676f02d9,14);

B:= C + LRot32(B + (D xor (A and (C xor D))) + Data[12] + $8d2a4c8a,20);

A:= B + LRot32(A + (B xor C xor D) + Data[ 5] + $fffa3942,4);

D:= A + LRot32(D + (A xor B xor C) + Data[ 8] + $8771f681,11);

C:= D + LRot32(C + (D xor A xor B) + Data[11] + $6d9d6122,16);

B:= C + LRot32(B + (C xor D xor A) + Data[14] + $fde5380c,23);

A:= B + LRot32(A + (B xor C xor D) + Data[ 1] + $a4beea44,4);

D:= A + LRot32(D + (A xor B xor C) + Data[ 4] + $4bdecfa9,11);

C:= D + LRot32(C + (D xor A xor B) + Data[ 7] + $f6bb4b60,16);

B:= C + LRot32(B + (C xor D xor A) + Data[10] + $bebfbc70,23);

A:= B + LRot32(A + (B xor C xor D) + Data[13] + $289b7ec6,4);

D:= A + LRot32(D + (A xor B xor C) + Data[ 0] + $eaa127fa,11);

C:= D + LRot32(C + (D xor A xor B) + Data[ 3] + $d4ef3085,16);

B:= C + LRot32(B + (C xor D xor A) + Data[ 6] + $04881d05,23);

A:= B + LRot32(A + (B xor C xor D) + Data[ 9] + $d9d4d039,4);

D:= A + LRot32(D + (A xor B xor C) + Data[12] + $e6db99e5,11);

C:= D + LRot32(C + (D xor A xor B) + Data[15] + $1fa27cf8,16);

B:= C + LRot32(B + (C xor D xor A) + Data[ 2] + $c4ac5665,23);

A:= B + LRot32(A + (C xor (B or (not D))) + Data[ 0] + $f4292244,6);

D:= A + LRot32(D + (B xor (A or (not C))) + Data[ 7] + $432aff97,10);

C:= D + LRot32(C + (A xor (D or (not B))) + Data[14] + $ab9423a7,15);

B:= C + LRot32(B + (D xor (C or (not A))) + Data[ 5] + $fc93a039,21);

A:= B + LRot32(A + (C xor (B or (not D))) + Data[12] + $655b59c3,6);

D:= A + LRot32(D + (B xor (A or (not C))) + Data[ 3] + $8f0ccc92,10);

C:= D + LRot32(C + (A xor (D or (not B))) + Data[10] + $ffeff47d,15);

B:= C + LRot32(B + (D xor (C or (not A))) + Data[ 1] + $85845dd1,21);

A:= B + LRot32(A + (C xor (B or (not D))) + Data[ 8] + $6fa87e4f,6);

D:= A + LRot32(D + (B xor (A or (not C))) + Data[15] + $fe2ce6e0,10);

C:= D + LRot32(C + (A xor (D or (not B))) + Data[ 6] + $a3014314,15);

B:= C + LRot32(B + (D xor (C or (not A))) + Data[13] + $4e0811a1,21);

A:= B + LRot32(A + (C xor (B or (not D))) + Data[ 4] + $f7537e82,6);

D:= A + LRot32(D + (B xor (A or (not C))) + Data[11] + $bd3af235,10);

C:= D + LRot32(C + (A xor (D or (not B))) + Data[ 2] + $2ad7d2bb,15);

B:= C + LRot32(B + (D xor (C or (not A))) + Data[ 9] + $eb86d391,21);

Inc(CurrentHash[0],A);

Inc(CurrentHash[1],B);

Inc(CurrentHash[2],C);

Inc(CurrentHash[3],D);

Index:= 0;

FillChar(HashBuffer,Sizeof(HashBuffer),0);

end;

procedure Update(const Buffer; Size: longword);

var

PBuf: ^byte;

begin

Inc(LenHi,Size shr 29);

Inc(LenLo,Size\*8);

if LenLo< (Size\*8) then

Inc(LenHi);

PBuf:= @Buffer;

while Size> 0 do

begin

if (Sizeof(HashBuffer)-Index)<= DWord(Size) then

begin

Move(PBuf^,HashBuffer[Index],Sizeof(HashBuffer)-Index);

Dec(Size,Sizeof(HashBuffer)-Index);

Inc(PBuf,Sizeof(HashBuffer)-Index);

Compress;

end

else

begin

Move(PBuf^,HashBuffer[Index],Size);

Inc(Index,Size);

Size:= 0;

end;

end;

end;

procedure Final(var Digest);

begin

HashBuffer[Index]:= $80;

if Index>= 56 then Compress;

PDWord(@HashBuffer[56])^:= LenLo;

PDWord(@HashBuffer[60])^:= LenHi;

Compress;

Move(CurrentHash,Digest,Sizeof(CurrentHash));

Burn;

end;

begin

Init;

Update(s[1],Length(s));

Final(a);

result:='';

for i:=0 to 15 do

result:=result+IntToHex(a[i],2);

Burn;

end;

procedure TLDSfunc.CreateShotCut(SourceFile, ShortCutName,

SourceParams: String);

var

IUnk: IUnknown;

ShellLink: IShellLink;

ShellFile: IPersistFile;

tmpShortCutName: string;

WideStr: WideString;

i: Integer;

begin

IUnk := CreateComObject(CLSID\_ShellLink);

ShellLink := IUnk as IShellLink;

ShellFile := IUnk as IPersistFile;

ShellLink.SetPath(PChar(SourceFile));

ShellLink.SetArguments(PChar(SourceParams));

ShellLink.SetWorkingDirectory(PChar(ExtractFilePath(SourceFile)));

ShortCutName := ChangeFileExt(ShortCutName,'.lnk');

if fileexists(ShortCutName) then

begin

ShortCutName := copy(ShortCutName,1,length(ShortCutName)-4);

i := 1;

repeat

tmpShortCutName := ShortCutName +'(' + inttostr(i)+ ').lnk';

inc(i);

until not fileexists(tmpShortCutName);

WideStr := tmpShortCutName;

end

else

WideStr := ShortCutName;

ShellFile.Save(PWChar(WideStr),False);

end;

procedure TLDSfunc.CreateShotCut\_InPrograms(RelationPath, FullExeName,

SourceParams: String; Handle: THandle);

var

WorkTable, LinkName:String;

P:PItemIDList;

C:array [0..1000] of char;

begin

if SHGetSpecialFolderLocation(Handle,CSIDL\_PROGRAMS,p)=NOERROR then

begin

SHGetPathFromIDList(P,C);

WorkTable:=StrPas(C)+'\'+RelationPath;

end;

if not DirectoryExists(WorkTable) then

CreateDir(WorkTable,0);

LinkName := ExtractFileName(FullExeName);

LinkName := Copy(LinkName, 1, Length(LinkName)-Length(ExtractFileExt(LinkName)));

LinkName := LinkName+'.lnk';

if FileExists(WorkTable+'\'+ExtractFileName(LinkName)) then

DeleteFile(WorkTable+'\'+ExtractFileName(LinkName));

CreateShotCut(FullExeName, WorkTable+'\'+ExtractFileName(FullExeName),

SourceParams);

end;

procedure TLDSfunc.CreateShotCut\_OnDesktop(FullExeName,

SourceParams: String; Handle: THandle);

var

WorkTable, LinkName:String;

P:PItemIDList;

C:array [0..1000] of char;

begin

if SHGetSpecialFolderLocation(Handle,CSIDL\_DESKTOP,p)=NOERROR then

begin

SHGetPathFromIDList(P,C);

WorkTable:=StrPas(C);

end;

LinkName := ExtractFileName(FullExeName);

LinkName := Copy(LinkName, 1, Length(LinkName)-Length(ExtractFileExt(LinkName)));

LinkName := LinkName+'.lnk';

if FileExists(WorkTable+'\'+ExtractFileName(LinkName)) then

DeleteFile(WorkTable+'\'+ExtractFileName(LinkName));

CreateShotCut(FullExeName, WorkTable+'\'+ExtractFileName(FullExeName),

SourceParams);

end;

function TLDSfunc.ExecuteFile(WND: HWND; const FileName, Params,

DefaultDir: string; ShowCmd: Integer): THandle;

var

zFileName, zParams, zDir: array[0..79] of Char;

begin

Result := ShellExecute(WND,nil,

StrPCopy(zFileName, FileName),StrPCopy(zParams, Params),

StrPCopy(zDir, DefaultDir), ShowCmd);

end;

function TLDSfunc.CreateRgnFromBitmap(Handle: THandle;

BmpFileName: TFileName): HRGN;

var

TransColor: TColor;

i, j: Integer;

i\_width, i\_height: Integer;

i\_left, i\_right: Integer;

rectRgn: HRGN;

rgnBitmap: TBitmap;

begin

Result := 0;

rgnBitmap:=TBitmap.Create;

rgnBitmap.LoadFromFile(BmpFileName);

i\_width := rgnBitmap.Width;

i\_height := rgnBitmap.Height;

transColor := rgnBitmap.Canvas.Pixels[0, 0];

for i := 0 to i\_height - 1 do

begin

i\_left := -1;

for j := 0 to i\_width - 1 do

begin

if i\_left < 0 then

begin

if rgnBitmap.Canvas.Pixels[j, i] <> transColor then

i\_left := j;

end

else

if rgnBitmap.Canvas.Pixels[j, i] = transColor then

begin

i\_right := j;

rectRgn := CreateRectRgn(i\_left, i, i\_right, i + 1);

if Result = 0 then

Result := rectRgn

else

begin

CombineRgn(Result, Result, rectRgn, RGN\_OR);

DeleteObject(rectRgn);

end;

i\_left := -1;

end;

end;

if i\_left >= 0 then

begin

rectRgn := CreateRectRgn(i\_left, i, i\_width, i + 1);

if Result = 0 then

Result := rectRgn

else

begin

CombineRgn(Result, Result, rectRgn, RGN\_OR);

DeleteObject(rectRgn);

end;

end;

end;

rgnBitmap.Free;

SetWindowRgn(Handle, Result, True);

end;

function TLDSfunc.GetOSInfo(OS\_typeinfo: Byte; Default: String): String;

var

OSVersion: TOSVersionInfo;

RegFile: TRegIniFile;

begin

Result := Default;

RegFile:=TRegIniFile.Create('Software');

OSVersion.dwOSVersionInfoSize:=SIZEOF(OSVersion);

if GetVersionEx(OSVersion) then

begin

if OS\_typeinfo=OS\_Version then // Получаем версию

Result:= Format('%d.%d (%d.%s)',[OSVersion.dwMajorVersion,

OSVersion.dwMinorVersion,(OSVersion.dwBuildNumber and $FFFF),

OSVersion.szCSDVersion]);

if OS\_typeinfo=OS\_Platform then // Получаем название платформы

case OSVersion.dwPlatformID of

VER\_PLATFORM\_WIN32s: Result := 'Windows 3.1';

VER\_PLATFORM\_WIN32\_WINDOWS: Result := 'Windows 95';{для 98-го то-же самое}

VER\_PLATFORM\_WIN32\_NT: Result := 'Windows NT';

else Result := '';

end;

end;

// Остальные данные получаем через реестр

RegFile.RootKey := HKEY\_LOCAL\_MACHINE;

RegFile.OpenKey('SOFTWARE', false);

RegFile.OpenKey('Microsoft', false);

RegFile.OpenKey('Windows', false);

case OS\_typeinfo of

OS\_Name: Result:= RegFile.ReadString('CurrentVersion','ProductName',Default);

OS\_Organization: Result:= RegFile.ReadString('CurrentVersion',

'RegisteredOrganization',Default);

OS\_Owner: Result:=RegFile.ReadString('CurrentVersion','RegisteredOwner',Default);

OS\_SerNumber: Result :=RegFile.ReadString('CurrentVersion','ProductId',Default);

OS\_WinPath: Result:=GetSystemPath(0, PATH\_WINDOWS);

OS\_SysPath: Result:=GetSystemPath(0, PATH\_SYSTEM);

OS\_TempPath: Result:=GetSystemPath(0, PATH\_TEMP);

OS\_ProgramFilesPath: Result:=GetSystemPath(0, PATH\_PROGRAMFILES);

OS\_IPName: Result := GetIPName;

end;

RegFile.CloseKey;

RegFile.Free;

end;

function TLDSfunc.GetSystemPath(Handle: THandle; PATH\_type: WORD): String;

var

P:PItemIDList;

C:array [0..1000] of char;

PathArray:array [0..255] of char;

begin

if PATH\_type = PATH\_TEMP then

begin

FillChar(PathArray,SizeOf(PathArray),0);

ExpandEnvironmentStrings('%TEMP%', PathArray, 255);

Result := Format('%s',[PathArray])+'\';

Exit;

end;

if PATH\_type = PATH\_WINDOWS then

begin

FillChar(PathArray,SizeOf(PathArray),0);

GetWindowsDirectory(PathArray,255);

Result := Format('%s',[PathArray])+'\';

Exit;

end;

if PATH\_type = PATH\_SYSTEM then

begin

FillChar(PathArray,SizeOf(PathArray),0);

GetSystemDirectory(PathArray,255);

Result := Format('%s',[PathArray])+'\';

Exit;

end;

if (PATH\_type = PATH\_PROGRAMFILES) or(PATH\_type = PATH\_COMMONFILES) then

begin

FillChar(PathArray,SizeOf(PathArray),0);

ExpandEnvironmentStrings('%ProgramFiles%', PathArray, 255);

Result := Format('%s',[PathArray])+'\';

if Result[1] = '%' then

begin

FillChar(PathArray,SizeOf(PathArray),0);

GetSystemDirectory(PathArray,255);

Result := Format('%s',[PathArray]);

Result := Result[1]+':\Program Files\';

end;

if (PATH\_type = PATH\_COMMONFILES) then Result := Result+'Common Files\';

Exit;

end;

if SHGetSpecialFolderLocation(Handle,PATH\_type,p)=NOERROR then

begin

SHGetPathFromIDList(P,C);

if StrPas(C)<>'' then

Result := StrPas(C)+'\' else Result:='';

end;

end;

function TLDSfunc.FindHexStringInFile(FileName, HexString: String;

StartByte: DWORD; TypeFind: Byte; Func: TFindProgress = nil): DWORD;

var

FS: TFileStream;

PosInFile: DWORD;

BufferArray: Array[1..8192] of Byte;

InputArray: Array[1..1000] of Byte;

InputArrayAdd: Array[1..1000] of Byte;

ReadSize: WORD;

InputArrayLength: WORD;

fSize, CurByte, I: DWORD;

ToEnd: DWORD;

StartByteToRead: DWORD;

C: WORD;

S1,S2: String;

begin

// Если TypeFind = 1, то будет HEX-последовательность

// Если TypeFind = 2, то будет искать текст по точному совпадению

// Если TypeFind = 3, то будет искать текст без учета регистра

Result := $FFFFFFFF; // Такой будет результат в случае неудачи

InputArrayLength := 0;

if (FileName = '') or (TypeFind < 1) or (TypeFind > 3) then Exit;

if not FileExists(FileName) then Exit;

// Если ищем HEX-последовательность

if TypeFind = 1 then begin

HexString := StringToHex(HexString);

if Length(HexString) mod 2 <> 0 then

Delete(HexString,Length(HexString),1);

if HexString='' then Exit;

InputArrayLength := Length(HexString) div 2;

for I := 1 to InputArrayLength do

InputArray[I]:=StrToInt('$'+Copy(HexString, I \* 2 - 1, 2));

end;

// Если ищем текст с учетом регистра

if (TypeFind = 2) then begin

if HexString='' then Exit;

InputArrayLength:=Length(HexString);

for I:=1 to InputArrayLength do

InputArray[I] := Ord(HexString[I]);

end;

// Если ищем текст без учета регистра

if (TypeFind = 3) then begin

if HexString='' then Exit;

InputArrayLength:=Length(HexString);

for I:=1 to InputArrayLength do begin

S1 := AnsiUpperCase(HexString[I]);

S2 := AnsiLowerCase(HexString[I]);

InputArray[I] := Ord(S1[1]);

InputArrayAdd[I] := Ord(S2[1]);

end;

end;

// Мы перевели входящие данные в массив данных, что облегчит дальнейшую

// обработку

fSize:=FileSize(FileName);

if fSize=0 then Exit;

PosInFile := StartByte;

C:=0;

// Открывает указанный файл для чтения

FS := TFileStream.Create(FileName, fmOpenRead,fmShareDenyWrite);

FS.Seek(StartByte, soFromBeginning);

while FS.Position < fSize do begin

if (FS.Position - InputArrayLength > PosInFile) then begin

StartByteToRead := FS.Position - InputArrayLength;

FS.Seek(StartByteToRead, soFromBeginning);

end;

ToEnd := fSize-FS.Position;

if ToEnd >= 8192 then ReadSize := 8192 else ReadSize := ToEnd;

PosInFile := FS.Position;

FS.Read(BufferArray, ReadSize);

Inc(C);

if C > 100 then begin

C := 0;

if @Func <> nil then

if not Func(FS.Position, FS.Size) then Break;

end;

CurByte := 0;

if TypeFind in [1, 2] then

while CurByte < ReadSize do begin

Inc(CurByte);

if (BufferArray[CurByte] = InputArray[1])

then begin

if InputArrayLength = 1 then begin

Result := FS.Position - (ReadSize - CurByte) - 1;

FS.Free;

Exit;

end;

for I := 2 to InputArrayLength do begin

if BufferArray[CurByte + I - 1] <> InputArray[I] then Break;

if I=InputArrayLength then begin

Result := FS.Position - (ReadSize - CurByte) - 1;

FS.Free;

Exit;

end;

end;

end;

end;

if TypeFind = 3 then

while CurByte < ReadSize do begin

Inc(CurByte);

if (BufferArray[CurByte] = InputArray[1]) or

(BufferArray[CurByte] = InputArrayAdd[1])

then begin

if InputArrayLength = 1 then begin

Result := FS.Position-(ReadSize-CurByte)-1;

FS.Free;

Exit;

end;

for I := 2 to InputArrayLength do begin

if (BufferArray[CurByte+I-1]<>InputArray[I]) and

(BufferArray[CurByte+I-1]<>InputArrayAdd[I])

then Break;

if I=InputArrayLength then begin

Result := FS.Position - (ReadSize - CurByte) - 1;

FS.Free;

Exit;

end;

end;

end;

end;

end;

FS.Free;

end;

function TLDSfunc.StringToHex(HexStr: String): String;

var

I: WORD;

HexSet: Set of '0'..'f' ;

begin

HexSet := ['0'..'9','a'..'f','A'..'F'];

if HexStr = '' then Exit;

// Отфильтровываем все знаки, оставляем только 16-ричные знаки

for I:=1 to Length(HexStr) do

if HexStr[I] in HexSet then Result := Result + HexStr[I];

end;

function TLDSfunc.GetProcessorInfo(CP\_typeinfo, CP\_NUM: Byte;

Default: String): String;

var

RegFile: TRegIniFile;

begin

Result := Default;

if CP\_typeinfo = CP\_SPEED then

begin

Result := IntToStr(Round(GetCPUSpeed));

Exit;

end;

RegFile:=TRegIniFile.Create('Software');

RegFile.RootKey := HKEY\_LOCAL\_MACHINE;

RegFile.OpenKey('hardware', false);

RegFile.OpenKey('DESCRIPTION', false);

RegFile.OpenKey('System', false);

RegFile.OpenKey('CentralProcessor', false);

case CP\_typeinfo of

CP\_Identifier :Result := RegFile.ReadString(IntToStr(CP\_NUM),

'Identifier', Default);

CP\_NameString :Result := RegFile.ReadString(IntToStr(CP\_NUM),

'ProcessorNameString', Default);

CP\_VendorIdentifier :Result := RegFile.ReadString(IntToStr(CP\_NUM),

'VendorIdentifier', Default);

CP\_MMXIdentifier :Result := RegFile.ReadString(IntToStr(CP\_NUM),

'MMXIdentifier', Default);

else Exit;

end;

RegFile.CloseKey;

RegFile.Free;

end;

function TLDSfunc.GetCPUSpeed: Double;

const

DelayTime = 500;

var

TimerHi, TimerLo: DWORD;

PriorityClass, Priority: Integer;

begin

PriorityClass := GetPriorityClass(GetCurrentProcess);

Priority := GetThreadPriority(GetCurrentThread);

SetPriorityClass(GetCurrentProcess, REALTIME\_PRIORITY\_CLASS);

SetThreadPriority(GetCurrentThread, THREAD\_PRIORITY\_TIME\_CRITICAL);

Sleep(10);

asm

dw 310Fh

mov TimerLo, eax

mov TimerHi, edx

end;

Sleep(DelayTime);

asm

dw 310Fh

sub eax, TimerLo

sbb edx, TimerHi

mov TimerLo, eax

mov TimerHi, edx

end;

SetThreadPriority(GetCurrentThread, Priority);

SetPriorityClass(GetCurrentProcess, PriorityClass);

Result := TimerLo / (1000.0 \* DelayTime);

end;

function TLDSfunc.GetIPName: String;

var

szHostName: array [0..255] of AnsiChar;

wsdata : TWSAData;

begin

WSAStartup ($0101, wsdata);

gethostname(szHostName, sizeof(szHostName));

Result := szHostName;

WSACleanup;

end;

initialization

LDSf := TLDSfunc.Create;

finalization

LDSf.Free;

end.