Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра «Информационная безопасность систем и технологий»

УТВЕРЖДАЮ

Зав. кафедрой ИБСТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Л. Зефиров

ОТЧЕТ

О КУРСОВОМ ПРОЕКТЕ

«Системное программирование для операционных систем»

Руководитель КП Н.А.Егорова

Исполнитель КП П.А. Пивоваров

Нормоконтролер Т.В. Щербакова

Пенза, 2007

# Реферат

Пояснительная записка содержит 39 с., 5 рис., 2 источника, 2 приложения.

Алгоритм, ДОКУМЕНТ, ПРИЛОЖЕНИЕ, ФАЙЛ, МАСТЕР, ОБЪЕКТ.

Целью проекта является разработка резидентной программы, контролирующую наличие одинаковых файлов на диске (в выбранных каталогах).

Требования к программе:

- программа должна иметь значок в System Tray, при закрытии ее главного окна программа должна оставаться в оперативной памяти, но соответствующая кнопка из панели задач должна быть удалена;

- при щелчке мышью на значке программы в System Tray должно вызываться всплывающее меню с командами для показа ее главного окна, и для ее закрытия (выгрузки из памяти);

- программа должна иметь возможность смены контролируемого каталога;

- программа должна иметь список имен файлов – исключений, а также средство для редактирования этого списка;

- программа должна в фоновом режиме контролировать состав файлов в контролируемом каталоге, и информировать пользователя о наличии/появлении одноименных файлов (с подготовкой отчета по одноименным файлам). В процессе работы была разработана программа, осуществляющая поиск, анализ содержимого каталога, написанная на языке программирования С++.

# Содержание

Реферат

Нормативные ссылки

Введение

1 Структура основной программы

2 Программная реализация алгоритма

3 Исходный текст программы приведен в приложении Б.

3 Функциональные возможности программы

4 Руководство по использованию программы

Заключение

Список использованных источников

Приложение Б (ОБЯЗАТЕЛЬНОЕ) Листинги программы

# Нормативные ссылки

В настоящем отчете использованы ссылки на следующие стандарты:

- ГОСТ 1.5 – 93 Государственная система стандартизации РФ. Общие требования к построению, изложению, оформлению и содержанию стандартов.

- ГОСТ 2.105 – 95 Единая система конструкторской документации.

- ГОСТ 2.301 – 68 ЕСКД Форматы.

- ГОСТ 7.1 – 84 Система стандартов по информации, библиотечному и издательскому делу. Библиографическое описание документа. Общие требования и правила составления.

- ГОСТ 7.32 – 2001 Система стандартов по информации, библиотечному, издательскому делу. Отчёт о научно – исследовательской работе.

# Введение

В настоящее время для подавляющего числа пользователей компьютер – это не роскошь, а средство работы с информацией, объем которой с каждым годом удваивается. Интернет, видео, музыка, документы гипертекста и так далее — все это служит для представления информации в нужном нам виде, обеспечивая тем самым довольно быстрый и удобный доступ к ней. А для создания всего этого используются языки программирования различного уровня. Например, пакет Microsoft Visual Studio. NET поддерживает следующие языки программирования: Visual Basic, Visual C++, Visual С#, Visual J#. Работая с языком Visual C++, в ваших руках оказывается мощный, полезный и широко распространенный инструмент. При его помощи можно получить в распоряжение довольно сильную и удобную систему создания приложений для операционных систем семейства Windows. С его помощью созданы такие продукты, как Excel и Access. Этот язык также применяется при разработке управленческих информационных систем и систем целевого назначения, используемых для анализа деятельности предприятий и принятия решений в сфере управления бизнесом. И, конечно же, целые армии хакеров и не только хакеров используют C++ для создания инструментов, утилит, игр и шедевров мультимедиа. Знания языка C++, позволяют создавать не просто приложения, а приложения, работающие в разных операционных системах. Возможности этого языка практически не ограничены.

Для создания удобного интерфейса программы под Windows, удобней использовать объектно-ориентированный язык программирования. Именно поэтому для реализации курсового проекта используется язык Microsoft Visual C++ 6.0. Автоматизированное создание приложений включено в компилятор Microsoft Visual C++ 6.0 и называется MFC AppWizard. MFC AppWizard создает тексты приложений только с использованием библиотеки классов MFC (Microsoft Foundation Class library). Таким образом, изучив язык C++ и библиотеку MFC, можно создавать свои приложения. Поэтому, используя средства автоматизированного создания приложений, можно сократить время написания довольно сложных программ. Так, при знании всех необходимых функций библиотеки MFC и WinAPI для написания программы по заданию курсового проекта потребуется не больше пары минут. При этом вручную придётся написать лишь несколько вызовов функций, а всё остальное сделает MFC AppWizard, используя возможности ClassWizard.

Суть задачи, поставленная в данном курсовом проекте, состоит в написании программы, которая должна осуществлять поиск одинаковых документов (по имени, размеру).

Пояснительная записка должна отражать следующие этапы работы: разработку алгоритмов программы, разработку пользовательского интерфейса, программную реализацию проекта, отладку и тестирование разработанной программы, описание работы с программой (руководство пользователя).

# Структура основной программы

При запуске программы на экран выводится диалоговое окно, на котором заданы поля для введения необходимых данных для поиска, поле для введения пути, кнопка «Скрыть окно», поля для вывода результата поиска.

Описания функций модуля FMain:

|  |  |
| --- | --- |
| Имя процедуры | Определение, назначение, выполняемые функции,  используемые параметры |
| TGlobalList::At | int \_\_fastcall TGlobalList::At(int Index)  Это метод списка строк TGlobalList, осуществляет доступ к свойству-массиву Objects, интерпретируя его элементы как тип (int) |
| TNameList::At | TIntArray \* \_\_fastcall TNameList::At(int Index)  Это метод списка строк TNameList, осуществляет доступ к свойству-массиву Objects, интерпретируя его элементы как тип (TIntArray \*) |
| TVerDirectory::TVerDirectory | TVerDirectory::TVerDirectory(TStrings \* fExNames, AnsiString fPath, TDirThread \* aOwner)  Это конструктор класса отчета. В качестве параметров принимает указатель на список строк-имен файлов исключений, путь к контролируемому каталогу, и указатель на родительский поток |
| TVerDirectory::~TVerDirectory | TVerDirectory::~TVerDirectory(void)  Это деструктор класса отчета. Выполняет действия по удалению объектов-полей класса |
| TVerDirectory::Clear | void \_\_fastcall TVerDirectory::Clear(void)  Этот метод класса отчета предназначен для сброса отчета в исходное состояние – выполняется очищение всех подготовленных списков для отчета (для списка ExNames удаляются только прицепленные массивы номеров каталогов, но не сами строки) |
| TVerDirectory::Fill | void \_\_fastcall TVerDirectory::Fill(void)  Главный метод класса отчета для проверки контролируемых каталогов и подготовки отчета. Если проверка была прервана в связи с изменением состава файлов (взведен DirectoryNever) - проверка начинается сначала. Если проверка была прервана в связи с завершением родительского потока - выход из функции. |
| TVerDirectory::WorkFile | bool \_\_fastcall TVerDirectory::WorkFile(int PathNum,AnsiString Name)  Проверка-регистрация в отчете файла (по имени и номеру подкаталога). При изменении состава файлов(взведен DirectoryNever) или при завершении родительского потока - досрочный выход с результатом - false |
| TVerDirectory::WorkDir | bool \_\_fastcall TVerDirectory::WorkDir(AnsiString Path)  Проверка-регистрация в отчете каталога (а также его содержимого, для подкаталогов - рекурсивные вызовы). При изменении состава файлов(взведен DirectoryNever) или при завершении родительского потока - досрочный выход с результатом - false |
| TVerDirectory::SetNameList | void \_\_fastcall TVerDirectory::SetNameList(TStrings \* DestList)  Метод класса отчета для формировки списка файлов-дубликатов в DestList |
| TVerDirectory::SetExNames | void \_\_fastcall TVerDirectory::SetExNames(TStrings \* DestList)  Метод класса отчета для формировки списка файлов-исключений в DestList |
| TVerDirectory::SetDirList | void \_\_fastcall TVerDirectory::SetDirList(int NameNum,TStrings \* DestList)  Метод класса отчета для формировки в DestList списка путей расположения файла номер NameNum (в списке одноименных файлов) |
| TVerDirectory::SetDirListEx | void \_\_fastcall TVerDirectory::SetDirListEx(int NameNum,TStrings \* DestList)  Метод класса отчета для формировки в DestList списка путей расположения файла номер NameNum (в списке файлов- исключений) |
| TVerDirectory::SetEx | bool \_\_fastcall TVerDirectory::SetEx(int Index)  Метод класса отчета для переноса файла-дубликата из списка одноименных файлов в список файлов-исключений. Index – индекс файла-дубликата в списке одноименных файлов |
| TVerDirectory::ResetEx | bool \_\_fastcall TVerDirectory::ResetEx(int Index)  Метод класса отчета для переноса имени файла из списка файлов-исключений в список одноименных файлов. Если файл с данным именем отсутствует в контролируемом каталоге, или присутствует в нем в единственном числе – пользователю в соответствующем сообщении предлагается просто удалить это имя из списка исключений. Index – индекс файла в списке файлов-исключений. |
| TDirThread::CheckStep | void \_\_fastcall TDirThread::CheckStep(void)  Метод потока TDirThread для включения мигания надписи "Запущена проверка" |
| TDirThread::SetLists | void \_\_fastcall TDirThread::SetLists(void)  Этот метод потока TDirThread передает подготовленный отчет главному окну программы (с открытием и активизацией главного окна) |
| TDirThread::Execute | void \_\_fastcall TDirThread::Execute()  Это главная функция потока проверки файлов TDirThread. Первую проверку выполняет безусловно, последующие проверки - по взводу события NeverEvent (от потока контроля файлов TNotifyThread) |
| TNotifyThread::Execute | void \_\_fastcall TNotifyThread::Execute()  Это главная функция потока автоматического контроля файлов каталогов TNotifyThread (переименование, добавление, удаление файлов) работает через механизм FindFirstChangeNotification - FindNextChangeNotification. Предусмотрен механизм завершения потока - по взводу события NeverEvent. |
| TFForm::TFForm | \_\_fastcall TFForm::TFForm(TComponent\* Owner)  Это конструктор формы TFForm |
| TFForm::NameBoxClick | void \_\_fastcall TFForm::NameBoxClick(TObject \*Sender)  Это обработчик выбора элемента списка NameBox (одноименные файлы). Обновляет содержимое списка путей расположения выбранного файла. |
| TFForm::ExBoxClick | void \_\_fastcall TFForm::ExBoxClick(TObject \*Sender)  Это обработчик выбора элемента списка ExBox (файлы-исключения). Обновляет содержимое списка путей расположения выбранного файла. |
| TFForm::CallThreads | void \_\_fastcall TFForm::CallThreads(void)  Это метод формы для создания и запуска потоков проверки и контроля файлов каталогов |
| TFForm::StopThreads | void \_\_fastcall TFForm::StopThreads(void)  Это метод формы для остановки и удаления потоков проверки и контроля файлов каталогов |
| TFForm::FormActivate | void \_\_fastcall TFForm::FormActivate(TObject \*Sender)  Обработчик OnActivate формы (работает только при запуске программы). Регистрирует значок в System Tray, инициализирует переменные, загружает из файла «FileNames.inf» (если он есть) путь к контролируемому каталогу и список файлов-исключений, и запускает потоки контроля и проверки. |
| TFForm::WMShellMess | void \_\_fastcall TFForm::WMShellMess(TMessage &Message)  Обработчик сообщения WM\_SHELLMESS от значка в System Tray - показывает всплывающее меню при щелчке мышью на значке в System Tray |
| TFForm::ShowMyWin | void \_\_fastcall TFForm::ShowMyWin(TMessage &Message)  Обработчик сообщения (WM\_USER+2) от второй копии приложения - для открытия и показа главного окна |
| TFForm::N1Click | void \_\_fastcall TFForm::N1Click(TObject \*Sender)  Обработчик пункта всплывающего меню "Отчет, настройки программы", открывает и показывает главное окно |
| TFForm::FormClose | void \_\_fastcall TFForm::FormClose(TObject \*Sender, TCloseAction &Action)  Обработчик OnClose (при закрытии) формы. При попытке закратия главного окна пиктограммой – только скрывает окно, а при выходе из программы (AppClose=true) – перед закрытием выполняет действия по освобождению памяти и занятых ресурсов, а также сохраняет параметры (контролируемый каталог и список исключений) в файл «FileNames.inf». |
| TFForm::N2Click | void \_\_fastcall TFForm::N2Click(TObject \*Sender)  Обработчик пуккта всплывающего меню "Закрыть программу", взводит признак AppClose и закрывает главное окно |
| TFForm::Button1Click | void \_\_fastcall TFForm::Button1Click(TObject \*Sender)  Обработчик кнопки "Скрыть окно" - скрывает главное окно |
| TFForm::Timer1Timer | void \_\_fastcall TFForm::Timer1Timer(TObject \*Sender)  Обработчик таймера - осуществляет мигание надписи "Запущена проверка..." |
| TFForm::AddDirButtonClick | void \_\_fastcall TFForm::AddDirButtonClick(TObject \*Sender)  Обработчик кнопки AddDirButton - открывает окно SDForm для смены контролируемого каталога, и перезапускает потоки |
| TFForm::SetExButtonClick | void \_\_fastcall TFForm::SetExButtonClick(TObject \*Sender)  Обработчик кнопки SetExButton (стрелка вниз) - передает файл из списка одноименных в список исключений |
| TFForm::ResetExButtonClick | void \_\_fastcall TFForm::ResetExButtonClick(TObject \*Sender)  Обработчик кнопки ResetExButton (стрелка верх) - передает файл из списка исключений в список одноименных файлов или удаляет файл из списка исключений |

# 

# Программная реализация алгоритма

При создании программы необходимо запустить среду программирования Microsoft Visual C++ 6.0.

Для использования MFC AppWizard необходимо выполнить следующие действия: в главном меню выбирается пункт File и подпункт New. В появившемся диалоговом окне нужно перейти на вкладку Project и выбрать MFCAppWizard. Затем в поле Project name нужно указать имя проекта (в данном курсовом проекте имя Crazzyff) и нажать кнопку Ok. Далее появится диалоговое окно первого шага MFC AppWizard Step1. В нём нужно указать тип создаваемого приложения Dialog based и нажать кнопку Next. На следующем шаге MFC AppWizard Step2 флажки на пунктах About box и 3D controls.В следующем окне MFC AppWizard Step3 требуется поставить переключатель на тип проекта MFC Standard и нажать кнопку Next. На четвёртом шаге можно просмотреть создаваемые классы и нажать кнопку Finish. Далее в окне New Project Information нужно нажать кнопку Ok. После этого проект будет создан и появится окно созданного проекта. В этом окне будут присутствовать следующие элементы: кнопки Ok и Cancel и статический текст. С помощью мыши можно изменить размер окна. Также можно переместить статический текст на требуемую позицию.

Список модулей проекта:

|  |  |
| --- | --- |
| FileNames.cpp | Главный модуль программы.  В сгенерированный автоматически код добавлена защита от запуска второй копии приложения. |
| ArrayTemplate.h  ArrayTemplate.cpp | Этот модуль реализует шаблон на динамические массивы со свойствами списка (коллекции). |
| FSelectDirForm.h  FSelectDirForm.cpp | Реализация формы выбора каталога. (код сгенерирован автоматически). |
| FMain.h  FMain.cpp | Реализация формы главного окна приложения.  Основная часть кода находится здесь. |

Описанные (вручную) типы:

|  |  |
| --- | --- |
| TArray | шаблон на динамический массив со свойствами списка элементов (коллекции) |
| TIntArray | динамический массив с элементами int  (на основе TArray) |
| THandleArray | динамический массив с элементами HANDLE (void \*)  (на основе TArray) |
| TGlobalList | тип-список строк для хранения имен файлов, а также номеров каталогов (в списке каталогов) для каждого имени |
| TNameList | тип-список строк для хранения имен файлов-дубликатов (файлов-исключений),а также указателей на массивы номеров каталогов (TIntArray \*) для каждого файла-дубликата |
| TDirThread | тип-поток для проверки каталогов и подготовки отчета по одноименным файлам |
| TNotifyThread | тип-поток для автоматического контроля проверяемых каталогов - при переименовке, добавлении и удалении файлов в данных каталогах (или их подкаталогах) инициирует проверку каталогов с подготовкой отчета |
| TVerDirectory | тип-подготавливаемый отчет по одноименным файлам, с набором методов для подготовки отчета. |

# Исходный текст программы приведен в приложении Б

# Функциональные возможности программы

Программа осуществляет поиск одинаковых документов по имени и размеру. В программе предусмотрена защита от запуска второй копии приложения. При запуске второй копии приложения – она открывает и активизирует главное окно первой копии, и завершает работу.

Необходимо отметить, что программа хранит свои параметры (путь к контролируемому каталогу и список исключений) в файле «FileNames.inf» (в каталоге с исполняемым модулем программы). При запуске программы – она считывает параметры из этого файла, при закрытии программы параметры будут сохранены в этот файл. Если при запуске программы этот файл отсутствует – в качестве контролируемого назначается каталог с исполняемым модулем программы, а список исключений пуст.

Также нужно сказать о том, что при запуске программы, при условии что одноименных файлов нет – ее главное окно будет скрыто через две секунды после подготовки первого отчета. Это сделано для того, чтобы программу можно было включать в меню автозагрузки Windows, как любую другую фоновую программу (при запуске программы ее главное окно выступает в качестве заставки).

# Руководство по использованию программы

Для запуска курсового проекта необходимо запустить файл Filenames.exe.

- Программа предназначена для контроля одноименных файлов на диске в выбранных каталогах.

- Загрузочный модуль программы - FileNames.exe .

- ОС Windows 9X, Windows NT/2000/XP , требования к машине (ПЭВМ) накладываются операционной системой.

При загрузке программа регистрирует значок в System Tray (правый нижний угол экрана), и при закрытии ее главного окна программа остается в памяти. Для закрытия программы надо щелкнуть мышью на значке в в System Tray и во всплывающем меню выбрать «Закрыть программу».

Для показа главного окна надо выбрать пункт всплывающего меню «Отчет, настройки программы» (см. рисунок 1).

Наглядное изображение программы приведено на рисунке 1:

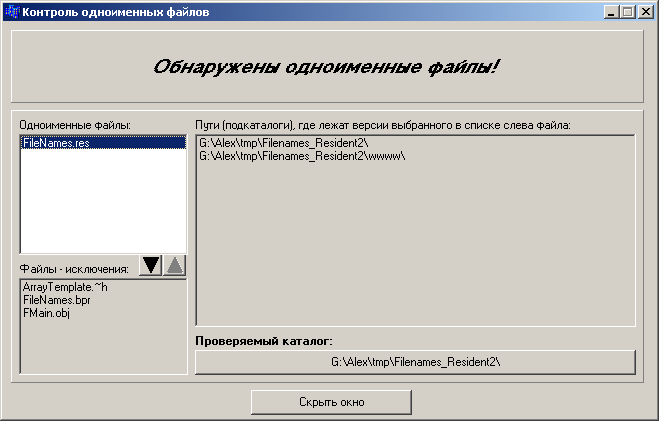


Рисунок 1 – Вид окна программы

Для смены контролируемого каталога нажмите кнопку под надписью «проверяемый каталог» (надпись на кнопке-путь к контролируемому каталогу).

В появившемся окне (рисунок 2) выберите нужный каталог и нажмите «Ввод»

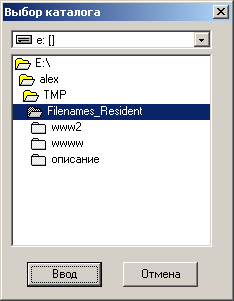


Рисунок 2 – Выбор каталога

При двойном щелчке на нужном каталоге окно выбора каталогов тоже закроется.

После этого будет произведена проверка (перепроверка) файлов для выбранного каталога (включая его подкаталоги), с подготовкой отчета по одноименным файлам.

При операциях добавления(копирования), удаления и переименования файлов в контролируемых каталогах – отчет по одноименным файлам обновляется автоматически, и если обнаруживаются одноименные файлы – будет выполнено оповещение пользователя (главное окно программы будет открыто и выведено на передний план).

Надо иметь в виду, что операция проверки выбранного каталога (с подготовкой отчета) может занять некоторое время, и сопровождается миганием надписи «Запущена проверка» в главном окне программы.

В главном окне программы также имеется список файлов-исключений, для которых оповещение пользователя не производится. Аналогично списку одноименных файлов, при выборе в списке исключений файла – в списке путей будут отображены пути расположения этого файла, или надпись «ФАЙЛ ОТСУТСТВУЕТ», если файл в контролируемом каталоге отсутствует.

Для переноса файла из списка одноименных файлов в список исключений выделите нужную строку в списке одноименных файлов и нажмите кнопку «перенести в список исключений».

Для переноса файла из списка исключений в список одноименных файлов выделите нужную строку в списке исключений и нажмите кнопку «перенести в список одноименных файлов». Если данный файл отсутствует в контролируемом каталоге, или присутствует в нем в единственном числе – в соответствующем сообщении вам будет предложено просто удалить данное имя из списка исключений.

# Заключение

# 

# В результате проделанной работы была создана программа, осуществляющая поиск одинаковых документов (по имени, размеру, дате создания). Во время проведения работы были приобретены навыки работы с языком программирования высокого уровня Visual C++. Исходя из того, что программа полностью соответствует заданию курсового проекта, можно сделать вывод, что задание на курсовую работу выполнено в полном объеме.

# Список использованных источников

1. Майкл Хаймен, Боб Арнсон. Visual C++: Учеб. Пособие. – М.: Диалектика, 2002. – 289 с.: ил.
2. А. Корера, С. Фрейзер, С. Маклин, Н. Кумар, С. Робинсон, П.Г. Саранг, С. Джентайл. Visual C++. Пособие для разработчиков. Изд-во «Лори», 2003. – 417 с.: ил.

# Приложение А

# (ОБЯЗАТЕЛЬНОЕ)

# Основной алгоритм программы

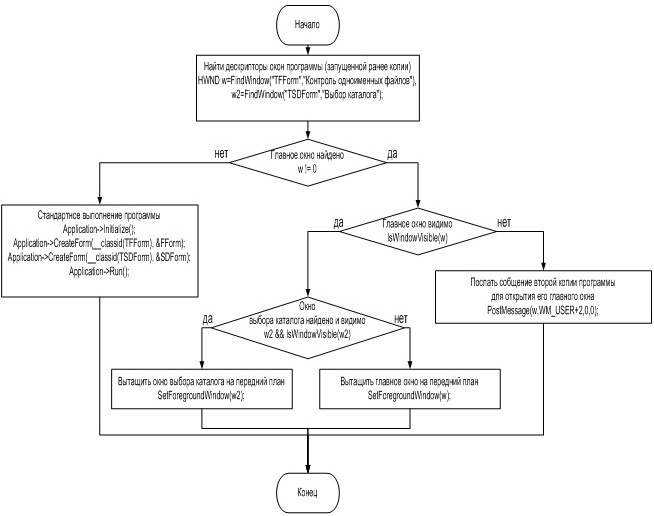


Рисунок А.1 – Основной алгоритм программы

Вышеуказанный алгоритм основной программы реализует защиту от запуска второй копии приложения. Если копия приложения уже запущена (найдены ее окна) – программа активизирует свою предыдущую копию и завершает работу, иначе – стандартное выполнение программы.

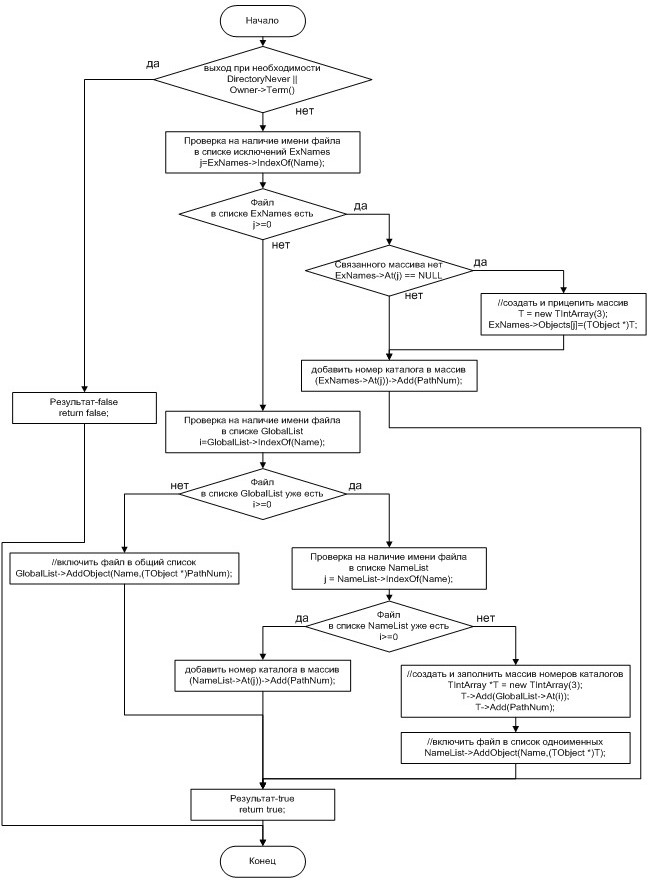


Рисунок А.2 – Алгоритм проверки регистрации по имени и номеру подкаталога

Вышеуказанный алгоритм реализует проверку и регистрацию файла в отчете. Этот метод отчета применяется к каждому найденному файлу в контролируемых каталогах.

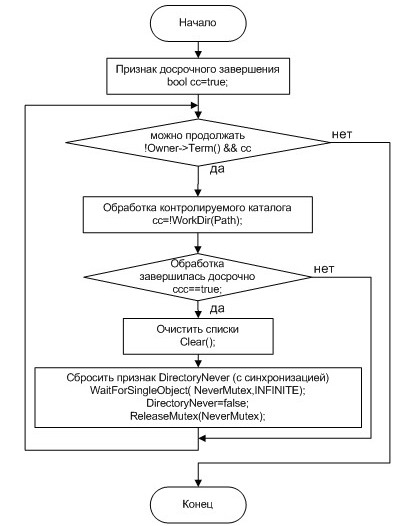


Рисунок А.3 – Главная функция для проверки каталогов и подготовки отчета

Вышеуказанный алгоритм реализует полный механизм проверки контролируемых каталогов и подготовки отчета. В алгоритме предусмотрены механизмы для досрочного завершения проверки по признаку завершения родительского потока, а также перезапуск проверки по признаку изменения файлов (от потока контроля файлов)

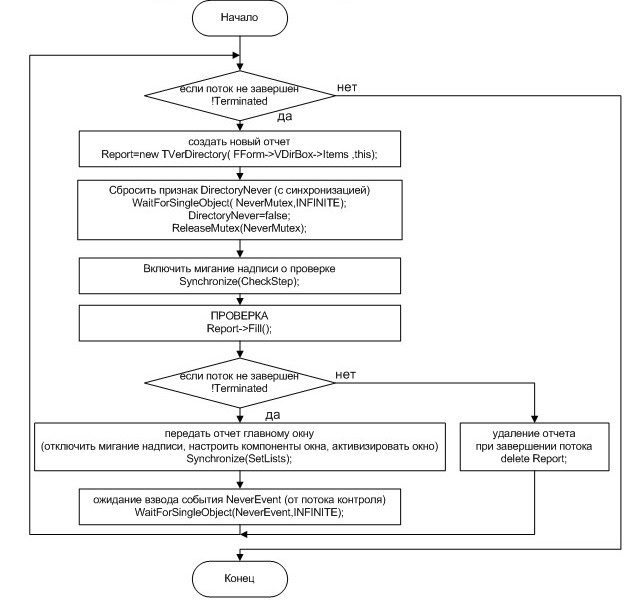


Рисунок А.4 – Главная функция потока проверки файлов

Вышеуказанный алгоритм реализует работу потока проверки файлов. Вызовы методов потока CheckStep и SetLists синхронизируются с главным VCL-потоком программы. Проверка файлов с передачей отчета главному окну программы инициируется взведением события NeverEvent - потоком контроля файлов по факту изменения состава файлов. Также предусмотрен механизм завершения потока – взведением свойства Terminated и события NeverEvent.

# Приложение Б

# (ОБЯЗАТЕЛЬНОЕ)

# Листинги программы

Файл проекта FileNames.cpp

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

//---------------------------------------------------------------------------

USEFORM("FMain.cpp", FForm);

USEFORM("FSelectDirForm.cpp", SDForm);

//---------------------------------------------------------------------------

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

{

try

{

//защита от повторного запуска

HWND w=FindWindow("TFForm","Контроль одноименных файлов"),

w2=FindWindow("TSDForm","Выбор каталога");

if(w) {//при обнаружении запущенной копии приложения - активизирует

//ее главное окно и завершает работу

if(IsWindowVisible(w)) {

if(w2 && IsWindowVisible(w2)) SetForegroundWindow(w2);

else SetForegroundWindow(w);

}

else PostMessage(w,WM\_USER+2,0,0);

}

else{ //иначе - стандартное выполнение программы

Application->Initialize();

Application->CreateForm(\_\_classid(TFForm), &FForm);

Application->CreateForm(\_\_classid(TSDForm), &SDForm);

Application->Run();

}

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

catch (...)

{

try

{

throw Exception("");

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

}

return 0;

}

//---------------------------------------------------------------------------

Файл ArrayTemplate.h

//---------------------------------------------------------------------------

#ifndef ArrayTemplateH

#define ArrayTemplateH

//---------------------------------------------------------------------------

//шаблон на динамический массив со свойствами коллекции(списка элементов)

template <class T> class TArray

{

private:

int fCount,fLength; //кол-во элементов,размер массива

T \*fItems; //указатель на массив

protected:

T \_\_fastcall Get(int Index); //чтение элемента массива

void \_\_fastcall Put(int Index, T Item); //запись элемента массива

void \_\_fastcall SetCount(int NewCount); //установка fCount

public:

TArray(int aLength); //конструктор

~TArray(void); //деструктор

void \_\_fastcall Insert(int Index,T Item);//вставка элемента

void \_\_fastcall Delete(int Index); //удаление элемента

void \_\_fastcall Add(T Item); //добавление элемента

void \_\_fastcall Clear(void); //удаление всех элементов

void \_\_fastcall SetLength (int NewLen); //определить размер массива

T& operator[](int Index); //оператор []-доступ к элементу

void\* operator&(void); //оператор & - адрес массива

\_\_property T Items[int Index] = {read=Get, write=Put}; //свойство для доступа к элементу

\_\_property int Count = {read=fCount, write=SetCount}; //свойство для доступа к кол-ву элементов списка

\_\_property int Length = {read=fLength, write=SetLength};//свойство для доступа к размеру массива

};

template <class T> TArray<T>::TArray(int aLength)

{

fCount=0;

fLength=0;

SetLength(aLength);

}

template <class T> TArray<T>::~TArray(void) { SetLength(0); }

template <class T> T& TArray<T>::operator[](int Index) { return fItems[Index];}

template <class T> void\* TArray<T>::operator&(void) {return fItems;}

template <class T> T \_\_fastcall TArray<T>::Get(int Index) { return fItems[Index]; }

template <class T> void \_\_fastcall TArray<T>::Put(int Index, T Item) { fItems[Index]=Item; }

template <class T> void \_\_fastcall TArray<T>::Add(T Item){ Insert(fCount,Item); }

template <class T> void \_\_fastcall TArray<T>::Clear(void) { fCount=0; }

template <class T> void \_\_fastcall TArray<T>::SetCount(int NewCount)

{

fCount=NewCount;

if(fCount<0) fCount=0;

if(fCount>fLength) fCount=fLength;

}

template <class T> void \_\_fastcall TArray<T>::Insert(int Index,T Item)

{

if(Index<0 || Index>fCount) return;

if (fCount==fLength) SetLength(fLength+5);

if (Index<fCount) Move(&fItems[Index],&fItems[Index+1],(fCount-Index)\*sizeof(T));

fItems[Index]=Item;

fCount++;

}

template <class T> void \_\_fastcall TArray<T>::Delete(int Index)

{

if(Index<0 || Index>fCount-1) return;

if (Index<fCount-1) Move(&fItems[Index+1],&fItems[Index],(fCount-1-Index)\*sizeof(T));

fCount--;

//if (fCount==fLength-6) SetLength(fCount+1);

}

template <class T> void \_\_fastcall TArray<T>::SetLength (int NewLen)

{

if (NewLen<0 || fLength==0 && NewLen==0) return; else

if (fLength==0) fItems=(T \*)SysGetMem(NewLen\*sizeof(T));else

if (NewLen==0) SysFreeMem(fItems);else fItems=(T \*)SysReallocMem(fItems,NewLen\*sizeof(T));

fLength=NewLen;

if (fLength<fCount) fCount=fLength;

}

//динамические массивы (с элементами int и HANDLE)

typedef TArray<int> TIntArray;

typedef TArray<HANDLE> THandleArray;

#endif

Файл ArrayTemplate.cpp

//---------------------------------------------------------------------------

#pragma hdrstop

#include <System.hpp>

#include "ArrayTemplate.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

Файл FSelectDirForm.h

//---------------------------------------------------------------------------

#ifndef FSelectDirFormH

#define FSelectDirFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <FileCtrl.hpp>

//---------------------------------------------------------------------------

class TSDForm : public TForm

{

\_\_published: // IDE-managed Components

TButton \*Button1;

TButton \*Button2;

TDirectoryListBox \*DirectoryListBox1;

TDriveComboBox \*DriveComboBox1;

private: // User declarations

public: // User declarations

\_\_fastcall TSDForm(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TSDForm \*SDForm;

//---------------------------------------------------------------------------

#endif

Файл FSelectDirForm.cpp

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "FSelectDirForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TSDForm \*SDForm;

//---------------------------------------------------------------------------

\_\_fastcall TSDForm::TSDForm(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

Файл FMain.h

//---------------------------------------------------------------------------

#ifndef FMainH

#define FMainH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include "ArrayTemplate.h"

#include <ExtCtrls.hpp>

#include <Menus.hpp>

#include <Buttons.hpp>

//---------------------------------------------------------------------------

//используемые коды сообщений

#define WM\_SHELLMESS (WM\_USER + 1) //от значка в System Tray

#define WM\_SHOWMYWIN (WM\_USER + 2) //от второй копии программы

//тип-список строк для хранения имен файлов, а также номеров каталогов

//(в списке каталогов) для каждого имени

class TGlobalList:public TStringList

{

public:

int \_\_fastcall At(int Index); //доступ к номеру каталога для выбранного файла

};

//тип-список строк для хранения имен файлов-дубликатов,а также указателей

//на массивы номеров каталогов (TIntArray \*) для каждого файла-дубликата

class TNameList:public TStringList

{

public:

TIntArray \* \_\_fastcall At(int Index); //доступ к массиву номеров для выбранного файла

};

//предварительное объявление

class TVerDirectory;

//тип-поток для проверки каталогов и подготовки отчета по одноименным файлам

class TDirThread:public TThread

{

protected:

void \_\_fastcall Execute(); //главная функция потока

public:

TVerDirectory \*Report; //подготавливаемый отчет

\_\_fastcall TDirThread(bool CreateSuspended):TThread(CreateSuspended) {} //конструктор

void \_\_fastcall SetLists(void); //передача отчета главному окну программы

void \_\_fastcall CheckStep(void); //отмечает в главном окне начало проверки каталогов

bool Term(void) {return Terminated;} //возвращает Terminated(protected - свойство)

};

//тип-поток для автоматического контроля проверяемых каталогов -

// при переименовке, добавлении и удалении файлов в данных каталогах (или их подкаталогах)

// инициирует проверку каталогов с подготовкой отчета

class TNotifyThread:public TThread

{

protected:

void \_\_fastcall Execute(); //главная функция потока

public:

\_\_fastcall TNotifyThread(bool CreateSuspended):TThread(CreateSuspended) {}//конструктор

bool Term(void) {return Terminated;} //возвращает Terminated(protected - свойство)

};

//тип-подготавливаемый отчет

class TVerDirectory

{

public:

TDirThread \* Owner; //поток-владелец отчета - только указатель

AnsiString Path;

TStringList \*DirList; //список подкаталогов проверяемого каталога

TGlobalList \*GlobalList; //общий список имен файлов

TNameList \*NameList; //список обнаруженных одноименных файлов

TNameList \*ExNames; //проверяемые каталоги(пути) - только указатель

TVerDirectory(TStrings \* fExNames, AnsiString fPath, TDirThread \* aOwner);//-конструктор

~TVerDirectory(void); //-деструктор

void \_\_fastcall Clear(void); //-очистка отчета

void \_\_fastcall Fill(void); //-главная функция для подготовки отчета

bool \_\_fastcall WorkFile(int PathNum,AnsiString Name); //-проверка файла

bool \_\_fastcall WorkDir(AnsiString Path); //-проверка каталога

// (с рекурсивными вызовами для подкаталогов)

void \_\_fastcall SetNameList(TStrings \* DestList); //-для передачи в оконный компонент списка одноименных файлов

void \_\_fastcall SetExNames(TStrings \* DestList); //-для передачи в оконный компонент списка исключений

void \_\_fastcall SetDirList(int NameNum,TStrings \* DestList);//-для передачи в оконный компонент списка путей расположения

// выбранного файла (с номером NameNum в списке NameList)

void \_\_fastcall SetDirListEx(int NameNum,TStrings \* DestList);//-для передачи в оконный компонент списка путей расположения

// выбранного файла (с номером NameNum в списке ExNames)

bool \_\_fastcall SetEx(int Index); //передать файл из NameList в ExNames

bool \_\_fastcall ResetEx(int Index); //передать файл из ExNames в NameList

};

//---------------------------------------------------------------------------

//тип - форма главного окна

class TFForm : public TForm

{

\_\_published: // IDE-managed Components

TButton \*Button1;

TPopupMenu \*PopupMenu1;

TMenuItem \*N1;

TMenuItem \*N2;

TPanel \*Panel1;

TLabel \*Label1;

TListBox \*NameBox;

TLabel \*Label2;

TListBox \*DirBox;

TPanel \*Panel2;

TLabel \*Label3;

TButton \*AddDirButton;

TTimer \*Timer1;

TLabel \*Label5;

TListBox \*ExBox;

TLabel \*Label4;

TSpeedButton \*SetExButton;

TSpeedButton \*ResetExButton;

TTimer \*Timer2;

void \_\_fastcall NameBoxClick(TObject \*Sender);

void \_\_fastcall FormActivate(TObject \*Sender);

void \_\_fastcall N1Click(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

void \_\_fastcall N2Click(TObject \*Sender);

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall Timer1Timer(TObject \*Sender);

void \_\_fastcall ExBoxClick(TObject \*Sender);

void \_\_fastcall AddDirButtonClick(TObject \*Sender);

void \_\_fastcall SetExButtonClick(TObject \*Sender);

void \_\_fastcall ResetExButtonClick(TObject \*Sender);

private: // User declarations

void \_\_fastcall WMShellMess(TMessage &Message); //обработчик WM\_SHELLMESS

void \_\_fastcall ShowMyWin(TMessage &Message); //обработчик WM\_SHOWMYWIN

public: // User declarations

TVerDirectory \*Report; //подготовленный отчет по одноименным файлам

TDirThread \*DirThr; //поток проверки каталогов

TNotifyThread \*NotifyThr; //поток автоматического контроля каталогов

TNotifyIconData NotifyData; //структура для регистрации значка в System Tray

bool AppClose; //признак выхода из программы

\_\_fastcall TFForm(TComponent\* Owner);//конструктор

void \_\_fastcall CallThreads(void); //создание и запуск потоков контроля и проверки каталогов

void \_\_fastcall StopThreads(void); //завершение и удаление потоков контроля и проверки каталогов

BEGIN\_MESSAGE\_MAP

MESSAGE\_HANDLER(WM\_SHELLMESS, TMessage, WMShellMess)

MESSAGE\_HANDLER(WM\_SHOWMYWIN, TMessage, ShowMyWin)

END\_MESSAGE\_MAP(TForm)

};

//---------------------------------------------------------------------------

extern PACKAGE TFForm \*FForm;

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#endif

Файл FMain.cpp

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "FMain.h"

#include "FSelectDirForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFForm \*FForm;

bool DirectoryNever; //признак необходимости перезапуска проверки каталогов

void \*NeverMutex, //мъютекс для синхронизации доступа к DirectoryNever (для записи)

\*NeverEvent; //событие для синхронизации потоков проверки файлов и контроля файлов

//---------------------------------------------------------------------------

//------------------------TGlobalList,TNameList---------------------------------

int \_\_fastcall TGlobalList::At(int Index) {return (int)Objects[Index];}

TIntArray \* \_\_fastcall TNameList::At(int Index) {return (TIntArray \*)Objects[Index];}

//---------------------------------------------------------------------------

//------------------------TVerDirectory----------------------------------------

TVerDirectory::TVerDirectory(TStrings \* fExNames, AnsiString fPath, TDirThread \* aOwner)

{

Path=fPath;

Owner=aOwner;

DirList= new TStringList();

GlobalList = new TGlobalList(); GlobalList->Sorted=true;

NameList = new TNameList(); NameList->Sorted=true;

ExNames = new TNameList(); ExNames->Sorted=true;

ExNames->AddStrings(fExNames);

}

//---------------------------------------------------------------------------

TVerDirectory::~TVerDirectory(void)

{

Clear();

delete DirList;

delete GlobalList;

delete NameList;

delete ExNames;

}

//---------------------------------------------------------------------------

void \_\_fastcall TVerDirectory::Clear(void)

{

for(int i=0; i<NameList->Count;i++) delete NameList->At(i);

for(int i=0; i<ExNames->Count;i++)

if (ExNames->At(i) != NULL)

{

delete ExNames->At(i);

ExNames->Objects[i]=NULL;

}

NameList->Clear();

GlobalList->Clear();

DirList->Clear();

}

//---------------------------------------------------------------------------

//главная функция для проверки каталогов и подготовки отчета

//если проверка была прервана в связи с изменением состава файлов (взведен DirectoryNever)-

//проверка начинается сначала.

//если проверка была прервана в связи с завершением родительского потока - выход из функции

void \_\_fastcall TVerDirectory::Fill(void)

{

bool cc=true;

while (!Owner->Term() && cc)

{

//проверка

cc=!WorkDir(Path);

if(cc)

{

Clear();

WaitForSingleObject( NeverMutex,INFINITE);

DirectoryNever=false;

ReleaseMutex(NeverMutex);

}

}

}

//---------------------------------------------------------------------------

//проверка-регистрация в отчете файла (по имени и номеру подкаталога)

//при изменении состава файлов(взведен DirectoryNever) или при завершении

//родительского потока - досрочный выход с результатом - false

bool \_\_fastcall TVerDirectory::WorkFile(int PathNum,AnsiString Name)

{

//выход при необходимости

if (DirectoryNever || Owner->Term() ) return false;

int i, j,k;

TIntArray \*T;

j=ExNames->IndexOf(Name);

if(j>=0)//имя в списке исключений

{

if(ExNames->At(j) == NULL) {//создать массив

T = new TIntArray(3);

ExNames->Objects[j]=(TObject \*)T;

}

(ExNames->At(j))->Add(PathNum);//добавить номер каталога

}

else

{

i=GlobalList->IndexOf(Name);

if(i>=0)//такой файл уже есть

{

j = NameList->IndexOf(Name);

if(j>=0) (NameList->At(j))->Add(PathNum);//добавить номер каталога в массив

else { //создать и заполнить массив номеров каталогов

T = new TIntArray(3);

T->Add(GlobalList->At(i));

T->Add(PathNum);

//включить файл в список одинаковых

NameList->AddObject(Name,(TObject \*)T);

}

}

else GlobalList->AddObject(Name,(TObject \*)PathNum);//включить файл в общий список

}

//выход

return true;

}

//---------------------------------------------------------------------------

//проверка-регистрация в отчете каталога (а также его содержимого, для подкаталогов - рекурсивные вызовы)

//при изменении состава файлов(взведен DirectoryNever) или при завершении

//родительского потока - досрочный выход с результатом - false

bool \_\_fastcall TVerDirectory::WorkDir(AnsiString Path)

{

//выход при необходимости

if (DirectoryNever || Owner->Term() ) return false;

TSearchRec SRec;

bool Result=true;

TStringList \*FList = new TStringList(),\*DList= new TStringList();

FList->Sorted=true;

DList->Sorted=true;

//заполнить списки файлов и каталогов

if(FindFirst(Path+"\*.\*",faAnyFile,SRec)==0)

do{

if(SRec.Name!="." && SRec.Name!="..")

{

if (SRec.Attr & faDirectory) DList->Add(SRec.Name);

else FList->Add(SRec.Name);

}

}while(FindNext(SRec)==0);

FindClose(SRec);

//обработка себя самого

DirList->Add(Path);

//обработка файлов

for(int i=0;i<FList->Count;i++)

if(!WorkFile(DirList->Count-1,FList->Strings[i])) {Result=false;break;}

delete FList;

//обработка подкаталогов

if(Result)

for(int i=0;i<DList->Count;i++)

if (!WorkDir(Path+DList->Strings[i]+"\\")) {Result=false;break;};

delete DList;

//выход

return Result;

}

//---------------------------------------------------------------------------

void \_\_fastcall TVerDirectory::SetNameList(TStrings \* DestList)

{

DestList->Clear();

for(int i=0;i<NameList->Count;i++) DestList->Add(NameList->Strings[i]);

}

//---------------------------------------------------------------------------

void \_\_fastcall TVerDirectory::SetExNames(TStrings \* DestList)

{

DestList->Clear();

for(int i=0;i<ExNames->Count;i++) DestList->Add(ExNames->Strings[i]);

}

//---------------------------------------------------------------------------

void \_\_fastcall TVerDirectory::SetDirList(int NameNum,TStrings \* DestList)

{

DestList->Clear();

for(int i=0;i<NameList->At(NameNum)->Count;i++)

DestList->Add(DirList->Strings[NameList->At(NameNum)->Items[i]] );

}

//---------------------------------------------------------------------------

void \_\_fastcall TVerDirectory::SetDirListEx(int NameNum,TStrings \* DestList)

{

DestList->Clear();

if (ExNames->At(NameNum) == NULL) DestList->Add("ФАЙЛ ОТСУТСТВУЕТ"); else

for(int i=0;i<ExNames->At(NameNum)->Count;i++)

DestList->Add(DirList->Strings[ExNames->At(NameNum)->Items[i]] );

}

//---------------------------------------------------------------------------

bool \_\_fastcall TVerDirectory::SetEx(int Index)

{

if(Index>=0 && Index<NameList->Count)

{

ExNames->AddObject(NameList->Strings[Index],NameList->Objects[Index]);

NameList->Delete(Index);

return true;

} else return false;

}

//---------------------------------------------------------------------------

bool \_\_fastcall TVerDirectory::ResetEx(int Index)

{

if(Index>=0 && Index<ExNames->Count)

{

if(ExNames->At(Index)==NULL)

{

if(Application->MessageBoxA(((AnsiString)"Выбранный файл отсутствует в контролируемом каталоге.\n"+

"Удалить его из списка исключений?").c\_str(),"Предупреждение",MB\_YESNO | MB\_ICONWARNING)==IDYES)

{

ExNames->Delete(Index);

return true;

}

} else

if(ExNames->At(Index)->Count==1)

{

if(Application->MessageBoxA(((AnsiString)"Выбранный файл в контролируемом каталоге присутствует в единственном числе.\n"+

"Удалить его из списка исключений?").c\_str(),"Предупреждение",MB\_YESNO | MB\_ICONWARNING)==IDYES)

{

delete ExNames->At(Index);

ExNames->Delete(Index);

return true;

}

} else {

NameList->AddObject(ExNames->Strings[Index],ExNames->Objects[Index]);

ExNames->Delete(Index);

return true;

}

}

return false;

}

//---------------------------------------------------------------------------

//------------------------------TDirThread--------------------------------------

//включает мигание надписи "Запущена проверка"

void \_\_fastcall TDirThread::CheckStep(void)

{

FForm->Timer1->Enabled=true;

FForm->Label5->Visible=true;

}

//---------------------------------------------------------------------------

//передает подготовленный отчет главному окну программы

//(с открытием и активизацией главного окна)

void \_\_fastcall TDirThread::SetLists(void)

{

//отключить мигание надписи "Запущена проверка"

FForm->Timer1->Enabled=false;

FForm->Label5->Visible=false;

TVerDirectory \*OldReport=FForm->Report;

bool First=(OldReport == NULL);//First=true при подготовке потоком первого отчета

FForm->Report=Report; //передача нового отчета

//настройка оконных компонентов под новый отчет

FForm->Report->SetNameList(FForm->NameBox->Items);

FForm->ExBox->ItemIndex=-1;

FForm->DirBox->Clear();

FForm->SetExButton->Enabled=false;

FForm->ResetExButton->Enabled=false;

if(Report->NameList->Count>0)

FForm->Panel2->Caption="Обнаружены одноименные файлы!";

else FForm->Panel2->Caption="Одноименные файлы отсутствуют.";

//открытие и активизация главного окна приложения

if(!First && (Report->NameList->Count>0 || OldReport->NameList->Count>0))

if(FForm->Visible) {

//Второй запуск самого себя - только такой путь, чтобы вытащить окно на передний

//план (SetForegroundWindow из другого процесса)

WinExec(ParamStr(0).c\_str(),SW\_SHOWNORMAL);

}

else {//открыть окно

ShowWindow(Application->Handle, SW\_SHOW);

FForm->Show();

}

if(!First) delete OldReport; //удалить старый отчет

//при подг-ке первого отчета (одноименных файлов нет) - скрыть окно через 2 сек.

else if (Report->NameList->Count==0) FForm->Timer2->Enabled=true;

}

//---------------------------------------------------------------------------

//главная функция потока проверки файлов

//первую проверку выполняет безусловно, последующие проверки - по взводу

//события NeverEvent (от потока контроля)

void \_\_fastcall TDirThread::Execute()

{

while(!Terminated)

{

//создать Report

Report=new TVerDirectory( FForm->ExBox->Items,FForm->AddDirButton->Caption ,this);

//сбросить DirectoryNever

WaitForSingleObject( NeverMutex,INFINITE);

DirectoryNever=false;

ReleaseMutex(NeverMutex);

//отметить начало проверки

Synchronize(CheckStep);

//проверка

Report->Fill();

if (!Terminated)//если поток не завершен

{

//передать отчет главному окну

Synchronize(SetLists);

//ожидание взвода события NeverEvent

WaitForSingleObject(NeverEvent,INFINITE);

}

else delete Report;//удаление отчета при завершении потока

}

}

//---------------------------------------------------------------------------

//------------------------------TNotifyThread--------------------------------------

//функция потока автоматического контроля файлов каталогов (переименование, добавление, удаление файлов)

//работает через механизм FindFirstChangeNotification - FindNextChangeNotification.

//предусмотрен механизм завершения потока - по взводу события NeverEvent

void \_\_fastcall TNotifyThread::Execute()

{

THandleArray Handles=THandleArray(2);

HANDLE Handle;

int i;

unsigned int Num;

Handles.Add(NeverEvent);

//поставить каталог на контроль, получить дескриптор ожидания

Handle=FindFirstChangeNotification( FForm->AddDirButton->Caption.c\_str(),

true,FILE\_NOTIFY\_CHANGE\_FILE\_NAME);

if (Handle!=INVALID\_HANDLE\_VALUE) Handles.Add(Handle);

while(!Terminated)

{

//ожидание событий (NeverEvent или Handles[1..Handles.Count-1])

Num=WaitForMultipleObjects(Handles.Count,&(Handles[0]),false,INFINITE);

if(Num==WAIT\_OBJECT\_0)//взвод события NeverEvent - предполагается завершение потока

{}else

if(Num>WAIT\_OBJECT\_0 && Num<WAIT\_OBJECT\_0+Handles.Count)

{ //взвод Handles[1..Handles.Count-1] - изменение файлов одного из каталогов

Num-=WAIT\_OBJECT\_0;//номер сработавшего HANDLE

Sleep(100);//подождать 100 мс

//взвести DirectoryNever (для перезапуска проверки, если проверка (в другом потоке) еще не закончена)

WaitForSingleObject( NeverMutex,INFINITE);

DirectoryNever=true;

ReleaseMutex(NeverMutex);

//взвести-сбросить событие NeverEvent (для запуска проверки в потоке проверки файлов)

PulseEvent(NeverEvent);

//снова поставить сработавший каталог на контроль

if(!FindNextChangeNotification(Handles[Num]))

{ //если на проверку не ставится

FindCloseChangeNotification(Handles[Num]);

Handles.Delete(Num);

}

}

}

//освободить дескрипторы ожидания

for(i=1;i<Handles.Count;i++) FindCloseChangeNotification(Handles[i]);

}

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

//-------------------------------TFForm--------------------------------------------

//конструктор формы

\_\_fastcall TFForm::TFForm(TComponent\* Owner)

: TForm(Owner)

{

Report=NULL;

AppClose=false;

}

//---------------------------------------------------------------------------

//обновляет содержимое списка DirBox при выборе файла в списке NameBox

void \_\_fastcall TFForm::NameBoxClick(TObject \*Sender)

{

if(Report && NameBox->ItemIndex>=0) Report->SetDirList(NameBox->ItemIndex,DirBox->Items);

SetExButton->Enabled=(NameBox->Items->Count>0 && NameBox->ItemIndex>=0);

ResetExButton->Enabled=(ExBox->Items->Count>0 && ExBox->ItemIndex>=0);

}

//---------------------------------------------------------------------------

//обновляет содержимое списка DirBox при выборе файла в списке ExBox

void \_\_fastcall TFForm::ExBoxClick(TObject \*Sender)

{

if(Report && ExBox->ItemIndex>=0) Report->SetDirListEx(ExBox->ItemIndex,DirBox->Items);

SetExButton->Enabled=(NameBox->Items->Count>0 && NameBox->ItemIndex>=0);

ResetExButton->Enabled=(ExBox->Items->Count>0 && ExBox->ItemIndex>=0);

}

//---------------------------------------------------------------------------

//создание запуск потоков проверки и контроля файлов каталогов

void \_\_fastcall TFForm::CallThreads(void)

{

DirectoryNever=false;

DirThr=new TDirThread(false);

NotifyThr=new TNotifyThread(false);

}

//---------------------------------------------------------------------------

//остановка и удаление потоков проверки и контроля файлов каталогов

void \_\_fastcall TFForm::StopThreads(void)

{

//взвести Terminated у обоих потоков

NotifyThr->Terminate();

DirThr->Terminate();

//взвести-сбросить событие NeverEvent для завершения потоков (для выхода из функций ожидания)

PulseEvent(NeverEvent);

DirThr->WaitFor();//дождаться завершения потока DirThr

delete DirThr; //удалить DirThr

NotifyThr->WaitFor();//дождаться завершения потока NotifyThr

delete NotifyThr; //удалить NotifyThr

//отключить мигание надписи "Запущена проверка"

Timer1->Enabled=false;

Label5->Visible=false;

}

//---------------------------------------------------------------------------

//обработчик OnActivate формы (работает только при запуске программы)

//регистрирует значок в System Tray, инициализирует переменные, запускает потоки

void \_\_fastcall TFForm::FormActivate(TObject \*Sender)

{

NotifyData.cbSize=sizeof(NotifyData);

NotifyData.hWnd=Handle;

NotifyData.uID=0;

NotifyData.uFlags=NIF\_ICON | NIF\_MESSAGE | NIF\_TIP;

NotifyData.uCallbackMessage=WM\_SHELLMESS;

NotifyData.hIcon=Application->Icon->Handle;

strncpy(&(NotifyData.szTip[0]),"Контроль одноименных файлов",sizeof(NotifyData.szTip));

Shell\_NotifyIcon(NIM\_ADD,&NotifyData);

Application->Title="Контроль файлов";

OnActivate=NULL;

NeverMutex=CreateMutex( NULL,false,"");

NeverEvent=CreateEvent( NULL,true,false,"");

if( FileExists(ChangeFileExt(ParamStr(0),".inf")) )//если есть файл параметров

{//прочитать файл параметров

TStringList \*S=new TStringList;

S->LoadFromFile(ChangeFileExt(ParamStr(0),".inf"));

AddDirButton->Caption=S->Strings[0]; //контролируемый каталог

for(int i=1;i<S->Count;i++) ExBox->Items->Add(S->Strings[i]);//список исключений

delete S;

}

else AddDirButton->Caption=ExtractFilePath(ParamStr(0));

SetExButton->Enabled=(NameBox->Items->Count>0 && NameBox->ItemIndex>=0);

ResetExButton->Enabled=(ExBox->Items->Count>0 && ExBox->ItemIndex>=0);

CallThreads();

}

//---------------------------------------------------------------------------

/\*

Msg.LParam=

512 - кнопки мыши не нажаты

513, 514 - нажать, отпустить левую кнопку ..001,..010

516, 517 - нажать, отпустить правую кнопку ..0100,..0101

519, 520 - нажать, отпустить среднюю кнопку ..0111,..1000

\*/

//обработчик событий от значка в System Tray -

//показывает всплывающее меню при щелчке мышью на значке в System Tray

void \_\_fastcall TFForm::WMShellMess(TMessage &Message)

{

if(Message.WParam ==0)

switch(Message.LParam)

{

case 513:

case 516:if(!SDForm->Visible)

{

TPoint tp;

SetForegroundWindow(Handle);

GetCursorPos(&tp);

N1->Enabled=!Visible || IsIconic(Application->Handle);

PopupMenu1->Popup(tp.x,tp.y);

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

}

break;

}

}

//---------------------------------------------------------------------------

//обработчик сообщения от второй копии приложения - для открытия и показа главного окна

void \_\_fastcall TFForm::ShowMyWin(TMessage &Message)

{

ShowWindow(Application->Handle, SW\_SHOWNORMAL);

Show();

}

//---------------------------------------------------------------------------

//обработчик пуккта всплывающего меню "Отчет, настройки программы"

//открывает и показывает главное окно

void \_\_fastcall TFForm::N1Click(TObject \*Sender)

{

ShowWindow(Application->Handle, SW\_SHOWNORMAL);

Show();

}

//---------------------------------------------------------------------------

//обработчик OnClose (при закрытии) формы

void \_\_fastcall TFForm::FormClose(TObject \*Sender, TCloseAction &Action)

{

Timer2->Enabled=false;

if(AppClose) {//закрытие программы разрешено - выполняет деинициализирующие действия

StopThreads();//остановка-удаление потоков

if(Report) delete Report; //удаление отчета

CloseHandle(NeverMutex); //освобождение объектов

CloseHandle(NeverEvent); //

Shell\_NotifyIcon(NIM\_DELETE,&NotifyData); //удаление значка из System Tray

//сохранение параметров в файл "FileNames.inf"

TStringList \*S = new TStringList();

S->Add(AddDirButton->Caption);//контролируемый каталог

for(int i=0;i<ExBox->Items->Count;i++) S->Add(ExBox->Items->Strings[i]);//список исключений

S->SaveToFile(ChangeFileExt(ParamStr(0),".inf"));

delete S;

}

else {//закрытие программы запрещено - только скрывает окно

Action=caNone;

Hide();

ShowWindow(Application->Handle, SW\_HIDE);

}

}

//---------------------------------------------------------------------------

//обработчик пуккта всплывающего меню "Закрыть программу"

//взводит признак AppClose и закрывает главное окно

void \_\_fastcall TFForm::N2Click(TObject \*Sender)

{

if(!SDForm->Visible)

{

AppClose=true;

Close();

}

}

//---------------------------------------------------------------------------

//обработчик кнопки "Скрыть окно" - скрывает главное окно

void \_\_fastcall TFForm::Button1Click(TObject \*Sender)

{

Timer2->Enabled=false;

Hide();

ShowWindow(Application->Handle, SW\_HIDE);

}

//---------------------------------------------------------------------------

//обработчик таймера - осуществляет мигание надписи "Запущена проверка..."

void \_\_fastcall TFForm::Timer1Timer(TObject \*Sender)

{

Label5->Visible=!Label5->Visible;

}

//---------------------------------------------------------------------------

//Обработчик кнопки AddDirButton -

//открывает окно SDForm для смены контр-го каталога, и перезапускает потоки

void \_\_fastcall TFForm::AddDirButtonClick(TObject \*Sender)

{

AnsiString Dir=AddDirButton->Caption;

if(Dir.Length() > 3) Dir.SetLength(Dir.Length()-1);

SDForm->DriveComboBox1->Drive=Dir[1];

SDForm->DirectoryListBox1->Directory=Dir;

if(SDForm->ShowModal()==mrOk)

{

Dir=SDForm->DirectoryListBox1->Directory;

if(Dir.Length() > 3) Dir=Dir+'\\';

StopThreads();

AddDirButton->Caption=Dir;

CallThreads();

}

}

//---------------------------------------------------------------------------

//Обработчик кнопки "Стрелка вниз" - передает файл из списка одноименных в список исключений

void \_\_fastcall TFForm::SetExButtonClick(TObject \*Sender)

{

if(Report->SetEx(NameBox->ItemIndex))

{

Report->SetNameList(NameBox->Items);

Report->SetExNames(ExBox->Items);

DirBox->Items->Clear();

SetExButton->Enabled=false;

ResetExButton->Enabled=false;

if(Report->NameList->Count>0) Panel2->Caption="Обнаружены одноименные файлы!";

else Panel2->Caption="Одноименные файлы отсутствуют.";

}

}

//---------------------------------------------------------------------------

//Обработчик кнопки "Стрелка верх" - передает файл из списка исключений в список одноименных

// или удаляет файл из списка исключений

void \_\_fastcall TFForm::ResetExButtonClick(TObject \*Sender)

{

if(Report->ResetEx(ExBox->ItemIndex))

{

Report->SetNameList(NameBox->Items);

Report->SetExNames(ExBox->Items);

DirBox->Items->Clear();

SetExButton->Enabled=false;

ResetExButton->Enabled=false;

if(Report->NameList->Count>0) Panel2->Caption="Обнаружены одноименные файлы!";

else Panel2->Caption="Одноименные файлы отсутствуют.";

}

}