Министерство образования и науки

Рязанский государственный радиотехнический университет

Кафедра ЭВМ

К защите

Руководитель проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ**

по дисциплине

**«Операционные системы»**

Тема

«Разработка, отладка и тестирование вспомогательной системной

программы в среде программирования Delphi с использованием средств Windows API»

Выполнил студент группы 342 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Маркина Полина Павловна дата сдачи на проверку, подпись

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Засорин Сергей Валентинович оценка дата защиты, подпись

Рязань 2015

Оглавление

[1. Введение 3](#_Toc136004336)

[2. Техническое задание 4](#_Toc136004337)

[2.1. Основания для разработки программы 4](#_Toc136004338)

[2.2. Назначение разработки 4](#_Toc136004339)

[2.3. Требования к программе 4](#_Toc136004340)

[2.4. Требования к надежности 4](#_Toc136004341)

[2.5. Требования к составу и параметрам технических средств 4](#_Toc136004342)

[2.6. Требования к программной документации 5](#_Toc136004343)

[3. Описание программы 6](#_Toc136004344)

[3.1. Общие сведения 6](#_Toc136004345)

[3.2. Функциональное назначение 6](#_Toc136004346)

[3.3. Описание логической структуры 6](#_Toc136004347)

[3.4.Создание справочной системы приложения. 13](#_Toc136004348)

[3.5. Создание инсталлятора 14](#_Toc136004349)

[3.6. Инсталлятор программы 14](#_Toc136004349)

[3.7. Используемые технические средства 17](#_Toc136004349)

[3.8. Вызов и загрузка 17](#_Toc136004349)

[3.9. Входные данные 17](#_Toc136004349)

[3.10. Выходные данные 12](#_Toc136004349)

[4. Пояснительная записка 19](#_Toc136004351)

[4.1. Введение 19](#_Toc136004352)

[4.2. Назначение и область применения 19](#_Toc136004353)

[4.3. Технические характеристики 20](#_Toc136004354)

[4.4. Ожидаемые технико-экономические показатели 20](#_Toc136004355)

[4.5. Источники, использованные при разработке 21](#_Toc136004356)

[5. Программа и методика испытаний 21](#_Toc136004357)

[5.1. Объект испытаний 21](#_Toc136004358)

[5.2. Цель испытаний 21](#_Toc136004359)

[5.3. Требования к программе 21](#_Toc136004360)

[5.4. Требования к программной документации 21](#_Toc136004361)

[5.5. Средства и порядок испытаний 22](#_Toc136004362)

[5.6. Методы и результаты испытаний 22](#_Toc136004363)

[5.6.1.Метод черного ящика 22](#_Toc136004364)

[5.6.2. Метод белого ящика 23](#_Toc136004365)

[6. Эксплуатационные документы 27](#_Toc136004366)

[6.1. Описание применения 27](#_Toc136004367)

[6.1.1. Назначение программы 27](#_Toc136004368)

[6.1.2. Условия применения 27](#_Toc136004369)

[6.1.3. Описание задачи 27](#_Toc136004370)

[6.1.4. Входные и выходные данные 27](#_Toc136004371)

[6.2.1. Общие сведения о программе 22](#_Toc136004372)

[6.2.3. Настройка программы 28](#_Toc136004373)

[6.2.4. Проверка программы 28](#_Toc136004374)

[6.2.5. Дополнительные возможности программы 28](#_Toc136004375)

[6.2.6. Сообщения системному программисту 28](#_Toc136004376)

[6.3. Руководство программиста 30](#_Toc136004377)

[6.3.1. Назначение и условия применения программы 30](#_Toc136004378)

[6.3.2. Характеристики программы 30](#_Toc136004379)

[6.3.3. Обращение к программе 24](#_Toc136004380)

[6.3.4. Входные и выходные данные 30](#_Toc136004381)

6.4 Код программы………………………………………………………………………………………………30

[7. Приложение 32](#_Toc136004382)

[7.1. Исходный текст модуля Unit11 (файл Unit11.pas) 32](#_Toc136004383)

[7.2. Исходный текст модуля Unit12 (файл Unit12.pas) 37](#_Toc136004384)

[7.3. Исходный текст модуля Unit4 (файл Unit4.pas) 46](#_Toc136004384)

[7.4. Исходный текст модуля Unit5 (файл Unit5.pas) 48](#_Toc136004384)

[8. Список литературы……………………………………………………………………………………… ……….50](#_Toc136004385)

# 1. Введение

Цель курсовой работы состоит в разработке двухпоточного приложения, выполняющего следующие операции:

1. Физическое копирование содержимого каталожной структуры с одного логического диска на другой.
2. Анализ расположения по отрезкам группы файлов на логическом диске.

# 2. Техническое задание

## 

## **2.1. Основания для разработки программы**

Основанием для разработки данной программы является задание на курсовую работу, вариант № 25.

## **2.2. Назначение разработки**

Программа предназначена для копирования содержимого произвольного каталога логического диска, копирования задаваемой каталожной структуры с любого диска компьютера на другой диск и вывода результата анализа расположения по отрезкам группы файлов на логическом диске.

Программа проста в использовании и рассчитана на средний уровень квалификации пользователя.

Программа предназначена для использования пользователями в среде операционных систем Windows /2000/XP/7/8/8.1/10.

## **2.3. Требования к программе**

Программа должна выполнять следующие функции:

1. Вычислить расположение отрезков группы файлов на диске-оригинале .
2. Скопировать содержимое каталога по его секторам на диск-копию.
3. Ввод исходных данных и вывод результатов должны выполняться через программный интерфейс.
4. В записке должна быть представлена полная документация на программу согласно ГОСТам ЕСПД, включая результаты тестирования.

## **2.4. Требования к надежности**

Программа гарантированно работает при соблюдении условий эксплуатации вычислительной системы, системного и прикладного программного обеспечения.

## **2.5. Требования к составу и параметрам технических средств**

* IBM PC совместимый компьютер;
* объем оперативной памяти не менее 32 Мб;
* видеокарта и VGA монитор;
* около 1 Мб свободной памяти на жестком магнитном диске (“винчестере”) для установки и запуска программы;
* мышь;
* клавиатура;
* операционная система Windows /2000/XP/7/Windows8.

## **2.6. Требования к программной документации**

Документация к программе представлена согласно требованиям следующими разделами:

1. Введение.

2. Техническое задание. (Раздел оформлен по ГОСТ 19.201-78).

3. Описание программы. (ГОСТ 19.402-78).

4. Пояснительная записка. (ГОСТ 19.404-79).

5. Программа и методика испытаний. (ГОСТ 19.301-79).

6. Эксплуатационные документы:

описание применения (ГОСТ 19.502-78);

руководство системного программиста (ГОСТ 19.503-79);

руководство программиста (ГОСТ 19.504-79).

7. Приложение.

8. Список литературы.

# 

# 3. Описание программы

## **3.1. Общие сведения**

Программа носит название “Курсовая работа вариант 25”. Программа служит для копирования содержимого произвольного каталога, анализа расположения по отрезкам группы файлов.

Программа разработана в среде программирования Borland Delphi 10 с применением стандартных компонент.

Для функционирования программы требуется операционная система Windows /2000/XP/7/8/8.1/10 с правами администратора. Файловые системы, поддерживаемые программой: NTFS.

## **3.2. Функциональное назначение**

Программа предназначена для анализа содержимого произвольного каталога логического диска, копирования задаваемой каталожной структуры по секторам с любого диcка компьютера на другой диск, а также для анализа расположения по отрезкам группы файлов. Оба эти действия могут выполняться по отдельности и параллельно друг другу.

## **3.3. Описание логической структуры**

Программа разрабатывалась в объектно-ориентированной среде программирования Borland Delphi 10. Справка разработана с помощью программы Help&Manual 6.2. В основном модуле программы создаются и запускаются два дополнительных потока.

Программа представлена следующими модулями:

1. Unit11 – основной модуль программы. В этом модуле осуществляется выбор каталога для осуществления его копирования и выбор группы файлов для анализа. Этот модуль обеспечивает взаимодействие программы и пользователя через программный интерфейс.
2. Unit12 – модуль предназначен для выполнения операции анализа содержимого каталога и копирования его содержимого. Этот модуль так же обеспечивает взаимодействие программы и пользователя через программный интерфейс.
3. Unit4-создает поток для реализации функции копирования.
4. Unit5-создает поток для реализации функции анализа.

Программа kr25 представлена следующими обязательными файлами:

1. kr25.dproj.local– файл проекта;
2. kr25.stat– сохранить статистику использования IDE для каждого проекта (время компиляции, отладки и т.д. )
3. kr25.res– файл ресурсов;
4. kr25.identcache– кэшированные привязки к файлам.
5. ABOUT.pas– файлы с текстом модуля ABOUT.
6. Unit11.pas – файлы с текстом модуля Unit11;
7. Unit12.pas – файлы с текстом модуля Unit12;
8. help.chm – файл проекта справки.
9. Unit4.pas – файлы с текстом модуля Unit4;
10. Unit5.pas – файлы с текстом модуля Unit5;

В программу после установки входят следующие файлы:

1. kr25.exe – исполняемый файл программы;
2. help.chm – файл справки;
3. help.hmxz – файл с заголовками для файла справки.

Общий алгоритм процедуры анализа отрезков файлов заданного каталога представлен на Рис.1.

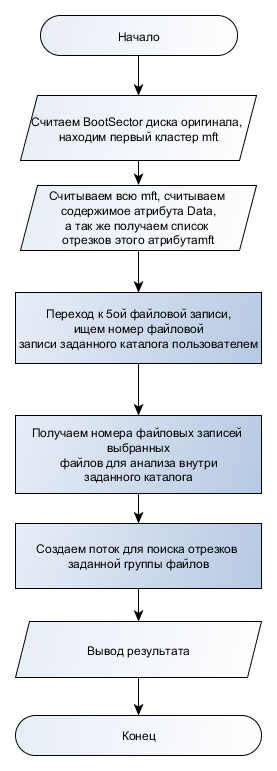


Рис.1. Общий алгоритм работы процедуры анализа.

В данной функции читается первый сектор логического диска-оригинала, который является загрузочным сектором(BootSector).Загрузочный сектор содержит в себе первый кластер Mft (*Master File Table* — «Главная файловая таблица»).Далее мы ищем атрибут Data с помощью процедуры Search(); Mft , который имеет заголовок 80h и содержит в себе список отрезков, чтобы получить список отрезков используем процедуру Unpuck();. Отрезки представляют собой последовательность кластеров, которые хранят файловые записи каталогов и файлов, хранящихся на диске. Далее берем номер файловой записи корневого каталога (который равен 5) и считываем содержимое данной файловой записи.

С помощью процедуры Search(); ищем атрибут Index Allocation(имеет заголовок 48h),так как кол-во файлов каталога превышает 3. Далее находим список отрезков Index Allocation с помощью процедуры GetFileRecordFile(); .Отрезки хранят индексные записи всех файлов и каталогов внутри каталога.Ищем среди индексных записей имя заданного каталога с помощью процедуры iaSearchDir ();,обнаружив совпадение между именем в индексной записи и именем каталога извлекаем номер файловой записи нужного нам каталога, однако если заданный каталог является подкаталогом, то прежде чем получить файловую запись этого каталога, придётся получить (файловые записи каталогов) файловую запись каталога в котором хранится наш каталог.

Получив номер файловой записи интересующего нас каталога, считываем ее содержимое. С помощью процедуры Search(); ищем атрибут Index Allocation(имеет заголовок 48h) ,далее извлекаем список отрезков процедурой GetFileRecordFile();.Следуя алгоритму изложенному выше, перебирая индексные записи(хранящихся в отрезках Index Allocation нашего каталога) ,с помощью процедуры iaSearchFile (); получим номера файловых записей группы выбранных файлов для анализа пользователем. Создаем поток для выполнения операции анализа,который содержит в себе операции описанные далее. Считываем атрибуты Data файлов и извлекаем из них отрезки. Отрезки представляют собой номера кластеров логического диска, в которых хранится содержимое файла.

Отрезок=номер первого кластера+смещение(кол-во кластеров в отрезке)=номер последнего кластера отрезка.

Далее представлены алгоритмы процедур, которые использовались для реализации анализа расположения отрезков группы файлов.

Search();

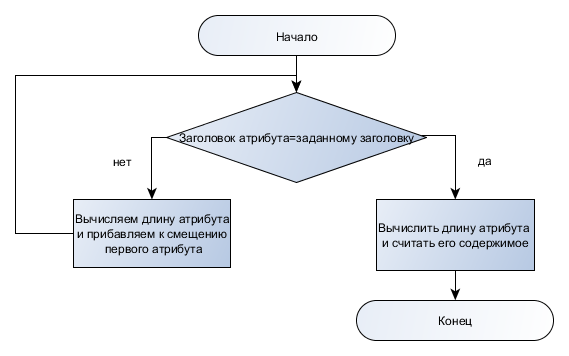


Рис.2. Общий алгоритм поиска нужного атрибута.

Unpuck();

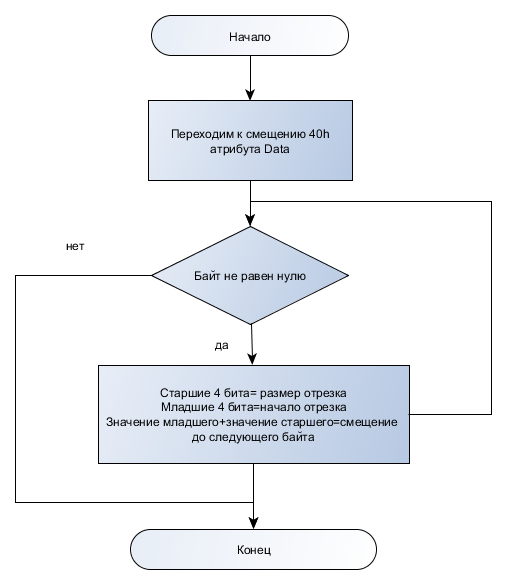


Рис.3. Общий алгоритм распаковки отрезков.

конец

Размер индексной записи+начало индексной записи

iaSearchDir ();

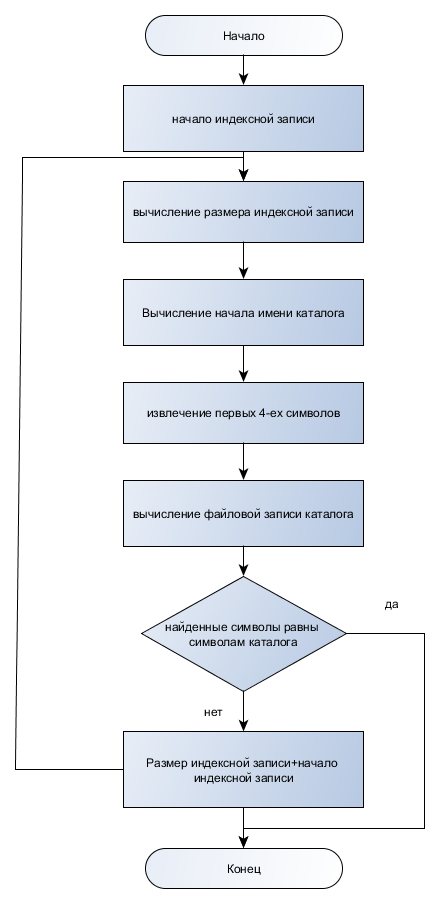


Рис.4. Общий алгоритм поиска имени каталога.

GetFileRecordDir();

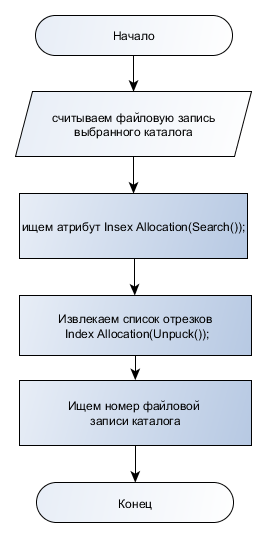


Рис.5. Общий алгоритм файловой записи каталога.

Примечание: алгоритм работы процедуры iaSearchDir ();аналогичен алгоритму процедуры iaSearchFile (); Различием являются лишь используемые переменные.

GetFileRecordDir(); так же аналогичен GetFileRecordFile (); .Различиями является ,что вместо GetFileRecordFile (); используется iaSearchFile ();,а не iaSearchDir ();.

После данных процедур создается поток ,который благодаря найденным номерам файловых записей группы файлов получает список отрезков файлов.

Операция копирования по секторам содержимого каталога полностью реализована в потоке. Копирование осуществляется с помощью рекурсивной функции DirSearch();,которая ищет подкаталоги внутри каталога до тех пор пока не окажется ,что вложенных каталогов больше нет. Если функция находит подкаталог, то она его создает на диске-копии реализует копирование файлов по секторам. Копирование осуществляется с помощью процедуры CopySector(); которая блоками равными в 512 байт (1 сектор) с помощью функции WinApi FileSeek (); переносит файлы с одного логического диска на другой.

**3.4. Создание справочной системы приложения.**

Справочная система разработана с помощью программы Help&Manual 6.2.

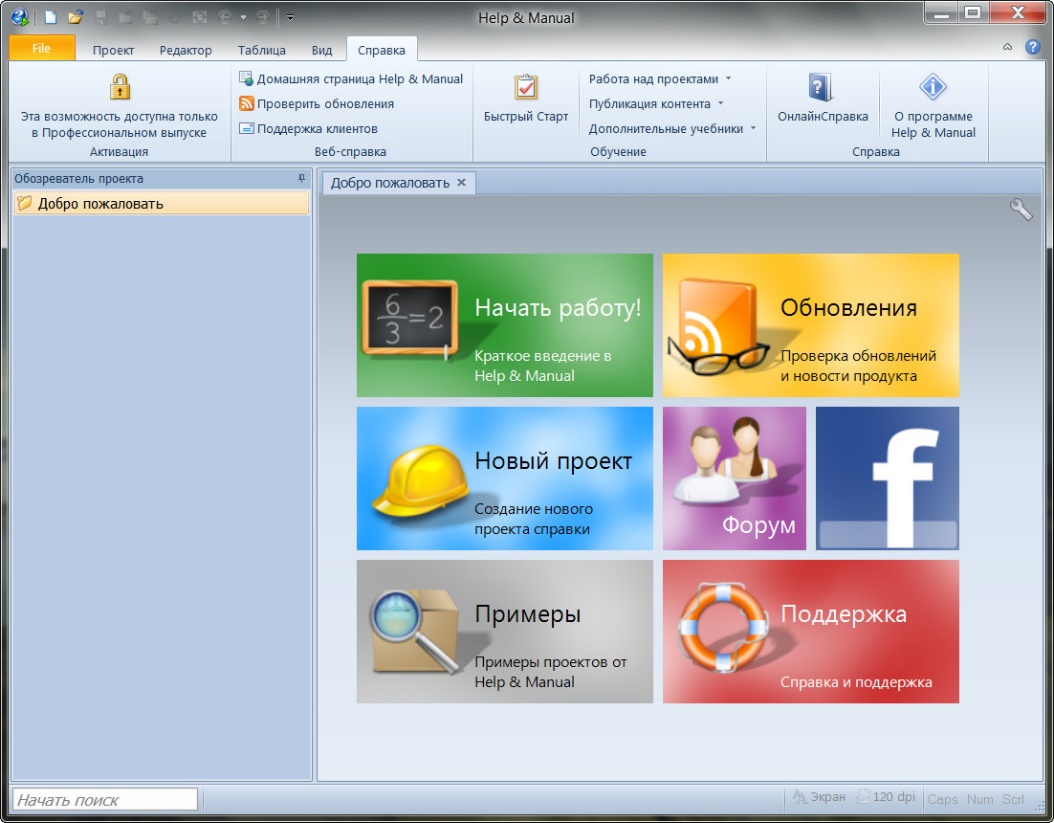


Рис.6. Окно проекта справочной системы.

Файл проекта справочной системы имеет расширение .hhp. После компиляции проекта справки получается файл с расширением .hhc.

Запустив и откомпилировав Help.hhc получаем файл Help.chm – файл справки:

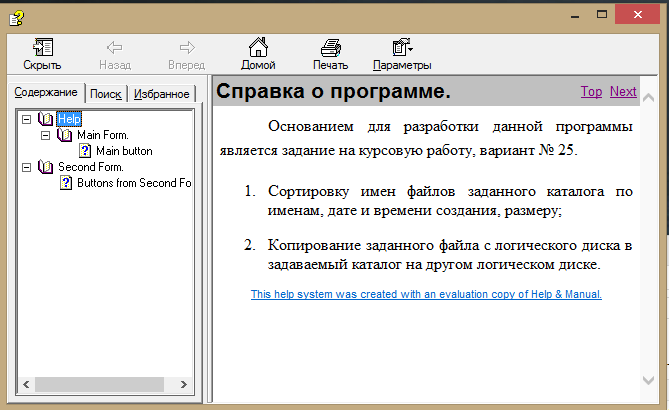


Рис.7. Окно справочной системы.

Справочная система подключается к программе следующим образом:

1. В обработчике события OnClick кнопки вызова справки указывается файл справки:

ShellExecute(0, 'open', Pchar(ExtractFilePath(Application.ExeName) + 'help.chm'), nil, nil, SW\_MAXIMIZE);

**3.5. Создание инсталлятора.**

Для создания инсталлятора используется программа Inno Setup:

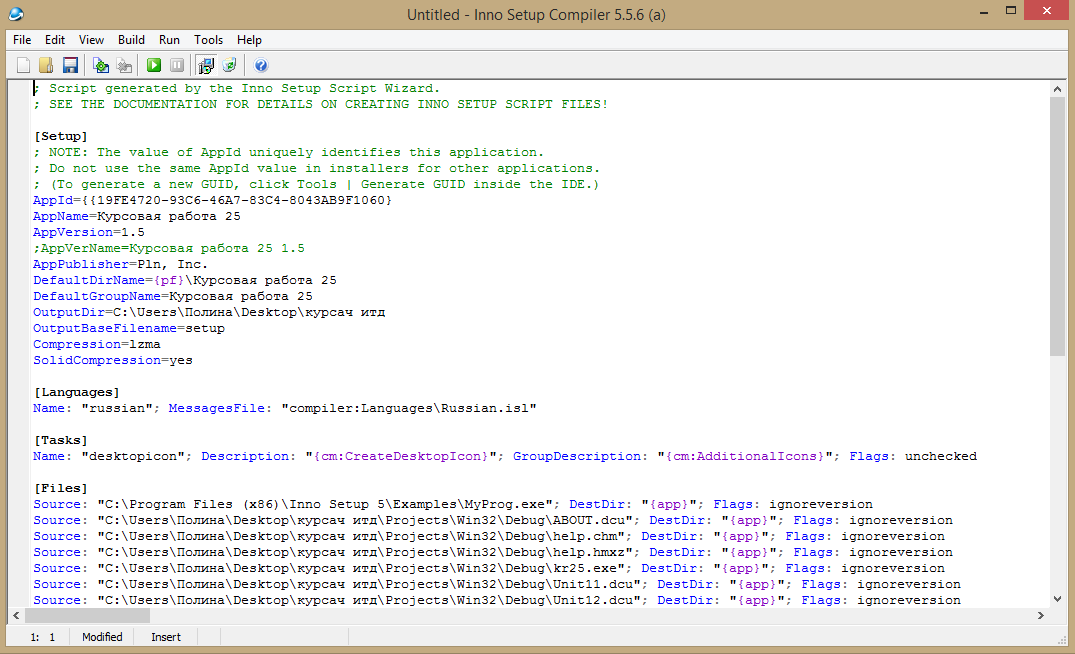


Рис.8. Окно проекта инсталлятора*.*

Указываем программе нужные параметры:

1. AppName – имя приложения;
2. AppVersion – версия приложения,
3. Директория для установки по умолчанию;
4. [Files] - файлы приложения для установки.

Указав параметры, компилируем проект, получаем файл Setup.exe – установщик проекта.

**3.6. Инсталляция программы.**

Для инсталляции программы запускаем файл Setup.exe.

Инсталляция проходит в несколько этапов, процесс инсталляции:

1. Выбор языка установки.

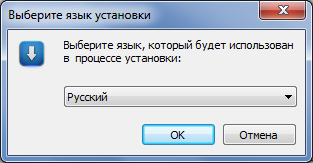


Рис.9. Окно выбора языка установки.

1. Ознакомление с именем и версией программы.

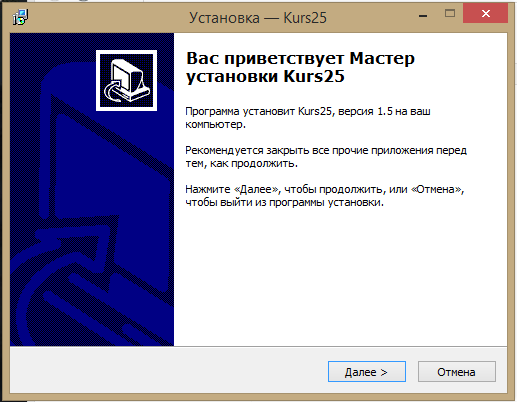


Рис.10. Окно ознакомления с именем и версией программы.

1. Выбор каталога для установки программы.

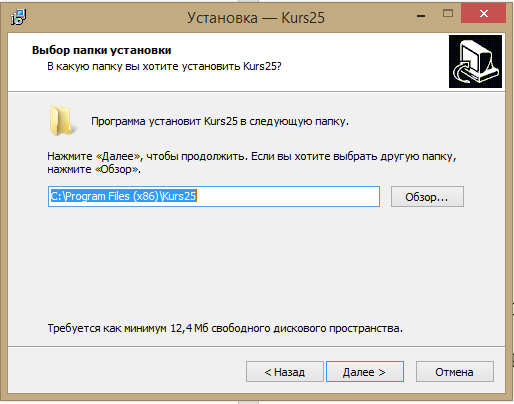
****

Рис.11. Окно выбора каталога для установки программы*.*

1. Выбор каталога для установки ярлыков программы.

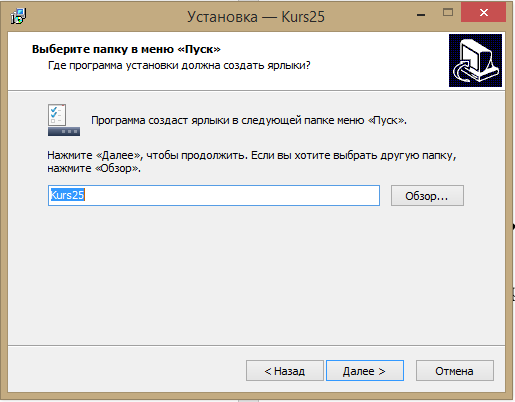


Рис.12. Окно выбора каталога для установки ярлыков программы.

1. Подтверждение установки

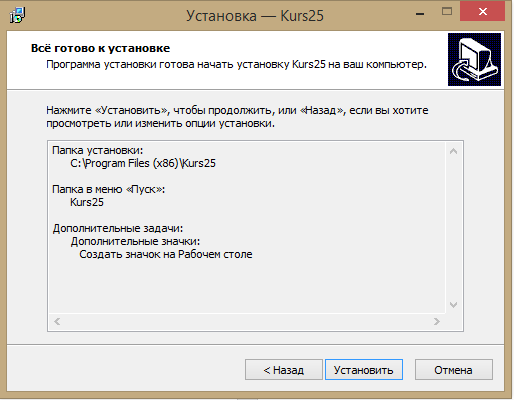


Рис.13. Окно завершения установки программы.

1. Завершение установки программы.

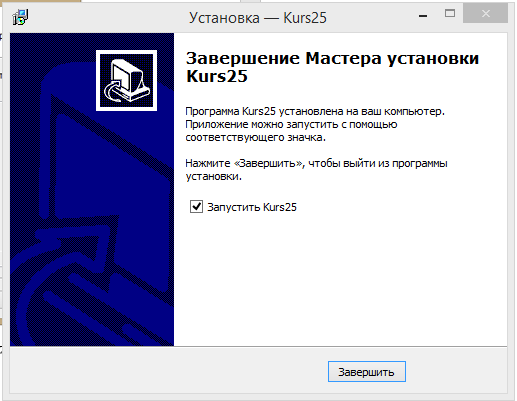


Рис.14. Окно завершения установки программы.

## **3.7. Используемые технические средства**

Для работы программы необходим персональный компьютер на базе процессора, совместимого с Intel® Pentium с установленной операционной системой Windows 2000/XP/7/8/8.1/10, монитор, клавиатура и/или манипулятор типа «мышь».

## **3.8. Вызов и загрузка**

Для установки программы на ПК ее следует установить с CD-R/RW, запустив файл Setup.exe. Программа установки предложит выбрать путь для инсталляции приложения.

Программа после установки может быть загружена двумя способами:

* запуском на исполнение файла kr25.exe, который находится в папке, указанной пользователем в процессе установки;
* из меню Пуск->kr25.

## **3.9. Входные данные**

При работе в качестве входных данных используется :

1. Имя логического диска.
2. Путь к задаваемой каталожной структуре.
3. Имена группы файлов для анализа.

## **3.10. Выходные данные**

Выходными данными в программе являются результаты анализа, выводящиеся на экран с возможностью сохранения в файл.

## **4. Пояснительная записка.**

## **4.1. Введение**

Программа носит название “ kr25”.

Основанием для разработки является задание на курсовую работу, вариант №25.

## **4.2. Назначение и область применения**

Программа предназначена для анализа расположения отрезков файлов произвольного каталога логического диска, а также для копирования по секторам содержимого задаваемой каталожной структуры с любого диска компьютера на другой диск в двухпоточном режиме и сортировки файлов заданного каталога по заданному признаку.

Для дополнительных сведений см. раздел 2.2.

## **4.3. Технические характеристики**

Поставлена задача разработки двухпоточного приложения с использованием функций Windows API, выполняющего следующие действия:

1.Анализ расположения отрезков выбранной группы файлов.

2. Копирование содержимого каталожной структуры с диска-оригинала на диск-копию.

Также ставится задача ввода исходных данных и вывода результатов через программный интерфейс, а также сохранение результатов в файле на диске. Для получения дополнительных сведений и информации о требованиях к программе и компоненте см. раздел 2.3.

При разработке программы использовался метод нисходящего проектирования, так как при этом возможно постепенное уточнение требований к разработке и реализуемому алгоритму.

В качестве входных данных, согласно заданию на курсовую работу (раздел 2.3), приняты:

1. Имя логического диска и путь к каталогу, содержащему файлы.
2. Полное имя каталога и имена файлов для анализа и копирования на новый диск.

Выходными данными в программе являются текстовые сообщения, выводимые в поле вывода на главной форме. Преимущества данного способа организации данных:

1. Информативность
2. Наглядность
3. Краткость

Необходимые для работы программы технические средства описаны в разделе 3.4.

Для программы kr25 разработана справочная система при помощи программного продукта Help&Manual 6.2.

В ходе работы программы kr25 справка может быть вызвана 3 способами:

1. Нажатие кнопки «Help».
2. Непосредственный запуск файла справки help2.hlp из каталога, куда была установлена программа.

Выбор в качестве программных средств объектно-ориентированной среды программирования Delphi версии 10, функций WinApi, входящих в ее состав, и программы установки InstallShield Express основан на требованиях задания на курсовую работу.

## **4.4. Ожидаемые технико-экономические показатели**

Программа обладает следующими свойствами:

1. не требует значительных затрат ресурсов и времени на разработку;
2. не требует дополнительного обучения пользователей для работы с ней;
3. реализует все требования, заложенные в техническом задании (раздел 2.3);
4. является простой в эксплуатации и обслуживании;
5. возможность применения программы на персональных компьютерах, имеющих только базовую конфигурацию системы.

## **4.5. Источники, использованные при разработке**

1. Методические указания к лабораторным работам по курсу «Операционные системы»/РГРТА Сост. Засорин С.В. Рязань, 2002, 56 с.
2. Архангельский А. Я. «Delphi 7. Справочное пособие»/Бином, Москва, 2003, 1024с.
3. Баженова И. Ю. «Delphi 7. Самоучитель программиста»/М, Кудиц-образ, 2003, 448с.
4. Шамис В. А. «Borland C++ Builder 6. Для профессионалов./СПб, Питер, 2004, 798с.
5. MSDN Library – May 2004.

# 5. Программа и методика испытаний

## **5.1. Объект испытаний**

Объектом испытаний являются программа “ kr25 ”, служащая для анализа содержимого произвольного каталога логического диска, копирования задаваемой каталожной структуры с любого диска компьютера на другой диск и сортировки файлов заданного каталога по заданному признаку.

## **5.2. Цель испытаний**

Целью испытаний является выявление возможных функциональных ошибок и сбоев, возникающих при работе данной программы, обеспечение их устойчивого функционирования в процессе эксплуатации и подтверждение работоспособности.

## **5.3. Требования к программе**

Необходимо проверить:

* правильность определения отрезков файла;
* правильность копирования содержимого каталожной структуры.

Программа должна функционировать:

* при задании любого каталога;
* на файловой системе NTFS;
* на любой ЭВМ, удовлетворяющей требованиям.

Программа должна предупреждать:

* об ошибках ввода/вывода;
* о неподдерживаемых файловых системах;
* об отсутствии выбранных файлов для анализа;

## **5.4. Требования к программной документации**

В программной документации к разработке должны присутствовать следующие разделы:

1. Введение (раздел 1).
2. Техническое задание (раздел 2).
3. Описание программы (раздел 3).
4. Пояснительная записка (раздел 4).
5. Программа и методика испытаний (раздел 5).
6. Эксплуатационные документы (раздел 6).
7. Приложение – исходные тексты с комментариями.
8. Список литературы.

## **5.5. Средства и порядок испытаний**

В процессе испытания использовалась конфигурация аппаратных средств, соответствующая требованиям к используемым техническим средствам (разделы 3.4, 4.3).

Компьютер с процессором Intel Core i5-3340 3.10 GHz;

Объём ОЗУ 8 Гб;

Видеоадаптер Nvidia GeForce GTX 760

Монитор Benq GL2760H.

Система программирования Borland Delphi 10 Seattle

Из программных средств использовались операционная система Microsoft Windows 8.1Professional, совместимая с требуемыми программными средствами (раздел 4.3).

Порядок испытаний:

1. Тестирование методом «чёрного ящика» (способом разбиения по эквивалентности) процедуры анализа расположения отрезков файлов. Результаты отражены в разделе 5.6.
2. Тестирование методом «белого ящика» процедуры копирования содержимого каталожной структуры. Результаты отражены в разделе 5.6

### **Метод «черного ящика»**

Тестирование процедуры копирования содержимого каталога будет проведено в соответствии с требованиями к разработке (раздел 2.3), методом «чёрного ящика» с использованием способа разбиения по эквивалентности.

Входными данными является имя каталога и имена выбранных файлов. Выделим следующие классы эквивалентности:

1. Каталог, содержащий подкаталоги и файлы.
2. Каталог, содержащий только файлы.
3. Не выбран ни один файл внутри каталога.

Правильность работы процедуры будем проверять с помощью стандартного приложения Windows “Проводник”.

В связи с тем, что имя каталога и имена файлов пользователь не вводит, а выбирает из списков, то некорректные исходные данные не возможны.

Результаты тестирования отразим в таблице 1.

Табл. 1. Результаты тестирования методом «чёрного ящика»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№**  **тестового задания** | **Описание тестового задания** | **Исходные данные** | **Ожидаемые результаты** | **Полученные результаты** |
| 1 | Каталог, содержащий подкаталоги и файлы. |  | Копирование успешно выполнено | Копирование каталога ,содержащего подкаталоги ,осуществлено |
| 2 | Каталог, содержащий только файлы . |  | Копирование успешно выполнен | Копирование, каталога содержащего только файлы осуществлено |
| 3 | Не выбран ни один файл внутри каталога. |  | Сообщение об ошибке |  |

Приведённые в таблице результаты позволяют сделать обоснованное предположение о правильности работы процедуры копирования и достоверности получаемых с её использованием результатов.

Отдельно было проведено тестирование визуальных элементов формы приложения kr25 (кнопок, главного меню, контекстного меню, всплывающих подсказок, строки состояния, главного окна, окон выбора диска, каталога, файла); все действия, выполняемые этими элементами отвечают их назначению, описанному в файле справки к программе.

### 

### **5.6.2. Метод «белого ящика»**

Тестирование процедуры ThrCopy.CopySector; копирования по секторам содержимого каталога.

Тестированием проводится с использованием внутренних возможностей среды разработки, и в частности окна **Watches.**

В этой процедуре происходит копирование файлов и подкаталогов указанного каталога. Ниже приводиться текст данной процедуры.

procedure ThrCopy.CopySector; //копирование по секторам

var

iFileHandle,iFileHandle1: Integer;

iFileLength: Integer;

iBytesRead: Integer;

Buffer: PChar;

i,j: Integer;

m:integer;

begin

I:=0;

Form12.Memo1.Lines.Add('Копирование файла:'+names);

SetCurrentDir(s1+'\'+CurrentName);

iFileHandle1:=CreateFile(Pchar(Names), GENERIC\_READ, FILE\_SHARE\_READ or FILE\_SHARE\_WRITE,nil,OPEN\_ALWAYS,0,0);

FileClose(iFileHandle1);

iFileHandle := FileOpen(s+'\'+CurrentName+'\'+Names, fmOpenRead);

iFileLength := FileSeek(iFileHandle,0,2);

iFileHandle1 := FileOpen(s1+'\'+CurrentName+'\'+Names, fmOpenWrite);

1: Buffer := PChar(AllocMem(512 + 1)); //копирование по блокам равным одному сектору

2: while I<=iFileLength do

3: begin

4: FileSeek(iFileHandle,0,I);

5: if ExitProcFlag then exit;

6: FileSeek(iFileHandle1,0,I);

iBytesRead := FileRead(iFileHandle, Buffer^, 512);

iBytesRead:=FileWrite(iFileHandle1, Buffer^, 512);

m:=I div 512;

Form12.Memo1.Lines.Add('Скопирован сектор:'+inttostr(m));

7: I:=I+512;

end;

FileClose(iFileHandle);

8: FileClose(iFileHandle1);

FreeMem(Buffer);

inc(j);

end;

Потоковый граф приведён на рисунке 1.



Рис. 8 Потоковый граф функции ThrCopy.CopySector;

Цикломатическая сложность графа может быть вычислена тремя способами:

1) V(G)=Число регионов потокового графа =3;

2)V(G)=E-N+2=9-8+2=3, где E – число дуг графа, N – количество его узлов;

3)V(G)=P+1=2+1=3, где Р – число предикатных узлов потокового графа.

Базовое множество путей приведено в таблице 2.

Таблица 2. Базовое множество путей потокового графа.

|  |  |
| --- | --- |
| **№ пути** | **Путь** |
|  | 1-2-3-4-5-7-8 |
|  | 1-2-3-4-6-7-8 |
|  | 1-2-3-4-6-7-1-2-3-4-6-7-1-2-3-4-6-7-1-2-3-4-6-7-8 |

Тестирование путей представлено в таблице 3.

Таблица 3. Результаты тестирования базового множества путей.

|  |  |  |  |
| --- | --- | --- | --- |
| Исходные данные | Путь | Результаты | |
| Полученные | Ожидаемые |
| poisk\_objekt (1).ppx | 1-2-3-4-5-7-8 | имя файла : poisk\_objekt (1).ppx  Копирование прервано! | имя файла : poisk\_objekt (1).ppx  Копирование прервано! |
| file.txt | 1-2-3-4-6-7-8 | имя файла : file.txt  Скопирован сектор:0  Копирование успешно завершено! | имя файла : file.txt  Скопирован сектор:0  Копирование успешно завершено! |
| Ewrewr.pdf | 1-2-3-4-6-7-1-2-3-4-6-7-1-2-3-4-6-7-1-2-3-4-6-7-8 | имя файла : Ewrewr.pdf  Скопирован сектор:1  Скопирован сектор:2.  Скопирован сектор:3  Скопирован сектор:4  Копирование успешно завершено! | имя файла : Ewrewr.pdf  Скопирован сектор:1  Скопирован сектор:2.  Скопирован сектор:3  Скопирован сектор:4  Копирование успешно завершено! |

При тестирование по всему множеству путей получены результаты, соответствующие ожидаемым. Это позволяет сделать вывод о правильной работе процедуры.

# 6. Эксплуатационные документы

## **6.1. Описание применения**

### **6.1.1. Назначение программы**

Программа предназначена для анализа содержимого произвольного каталога логического диска, копирования задаваемой каталожной структуры с любого диска компьютера на другой диск. Программа проста в использовании и рассчитана на средний уровень квалификации пользователя.

Программа предназначена для использования пользователями в среде операционных систем Windows 2000/XP/7/Windows8.

### **6.1.2. Условия применения**

Для работы программы необходим персональный компьютер на базе процессора, совместимого с Intel® Pentium с установленной операционной системой Windows 2000/XP/7/Windows8, монитор, клавиатура и/или манипулятор типа «мышь».

### **6.1.3. Описание задачи**

Программа выполняет следующие функции:

1. Вычисляет расположение отрезков группы файлов на диске.
2. Читает каталога по всем его секторам в память.
3. Сортирует файлы в заданном каталоге по заданному признаку.
4. Информацию о отрезках отображает на устройстве стандартного вывода или записать в файл.
5. Ввод исходных данных и вывод результатов выполняется через программный интерфейс.

### **6.1.4. Входные и выходные данные**

При работе, в качестве входных данных программа использует путь к подкаталогу для анализа и копирования. Выбор пути осуществляется с помощью визуальных компонент Delphi DriveComboBox, DirectoryListBox.

Выходные данные представляют собой информацию о ходе работы программы списка отрезков файлов . Они отображаются в поле вывода и могут быть сохранены в файл. Сохранить результаты работы программы в файле можно, нажав соответствующую кнопку.

### **6.2. Руководство системного программиста**

### **6.2.1. Общие сведения о программе**

Программа предназначена для анализа содержимого произвольного каталога логического диска и копирования задаваемой каталожной структуры с любого диска компьютера на другой диск.

Программа проста в использовании и рассчитана на средний уровень квалификации пользователя.

Программа предназначена для использования пользователями в среде операционных систем Windows 2000/XP/7/Windows8.

**6.2.2. Структура программы**

Рабочая версия программы состоит из следующих файлов:

1. kr25.exe – исполняемый файл программы;
2. help.hlp – файл справки;
3. help1.cnt – файл с заголовками для файла справки.
4. help1.gid – файл, создаваемый системой справки после первого запуска файла справки для хранения внутренней информации.

### **6.2.3. Настройка программы**

Программа kr25.exe после установки не нуждается в какой-либо дополнительной настройке или обслуживании.

### **6.2.4. Проверка программы**

В случае необходимости проведения дополнительного тестирования компоненты либо программы, рекомендуется опираться на результаты раздела 5.6 (Методы испытаний).

### **6.2.5. Дополнительные возможности программы**

Программа может быть использованы в составе более сложных средств предоставляющих различную информацию об расположении файлов на жёстких дисках.

### **6.2.6. Сообщения системному программисту**

1. «Выбор системного диска для анализа-запрещен» – возникает в случае прямого доступа к системному диску;

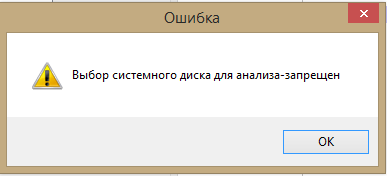


Рис. 9 Предупреждение об ошибке.

1. «Файловая система FAT не поддерживается» – возникает, если используется файловая система, отличная от NTFS; необходимо запустить программу в системе, с требуемой файловой системой.

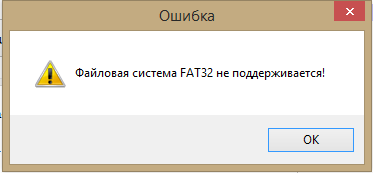


Рис. 10 Предупреждение об ошибке.

## **6.3. Руководство программиста**

### **6.3.1. Назначение и условия применения программы**

Программа предназначена для анализа расположения отрезков произвольного каталога логического диска и копирования задаваемой каталожной структуры с любого диска компьютера на другой диск. Оба эти действия могут выполняться по отдельности и параллельно друг другу.

### **6.3.2. Характеристики программы**

При работе, в качестве входных данных программа использует путь к каталогу для анализа и копирования. Выбор пути осуществляется с помощью визуальных компонент Delphi DriveComboBox, DirectoryListBox.

Выходные данные представляют собой информацию о ходе работы программы и цепочки кластеров найденных файлов. Они отображаются в поле вывода и могут быть сохранены в файл.

Программа имеет максимально удобный, дружественный интерфейс, предоставляя в распоряжение пользователя средства справочной системы, контекстной справки, различные подсказки и другие элементы, упрощающие работу.

Для программы спроектирована программа установки на основе генератора программ установки InstallShield.

### **6.3.3. Обращение к программе**

Для установки программы на ПК ее следует установить с CD-R/RW, запустив файл Setup.exe. Программа установки предложит выбрать путь для инсталляции приложения.

Программа после установки может быть загружена двумя способами:

* запуском на исполнение файла CAS.exe, который находится в папке, указанной пользователем в процессе установки;
* из меню Пуск->CAS.

### **6.3.4. Входные и выходные данные**

При работе, в качестве входных данных программа использует путь к каталогу для анализа и копирования. Выбор пути осуществляется с помощью визуальных компонент Delphi DriveComboBox, DirectoryListBox.

Выходные данные представляют собой информацию о ходе работы программы и цепочки кластеров найденных файлов. Они отображаются в поле вывода и могут быть сохранены в файл.

**6.3.5. Сообщения программисту**

В случае возникновения ошибок в работе программы, программисту будет выдано одно из следующих сообщений об ошибке:

1. «Выбор системного диска для анализа-запрещен» – возникает в случае прямого доступа к системному диску;

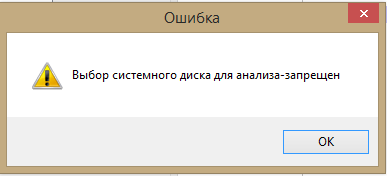


Рис. 11 Предупреждение об ошибке.

1. «Файловая система FAT не поддерживается» – возникает, если используется файловая система, отличная от NTFS; необходимо запустить программу в системе, с требуемой файловой системой.

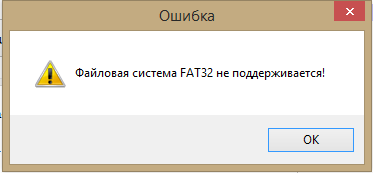


Рис. 12 Предупреждение об ошибке.

# 7. Приложение

## **7.1. Исходный текст модуля Unit11 (файл Unit11.pas)**

unit Unit11;

interface

uses

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

Dialogs, StdCtrls, FileCtrl,ComCtrls, Buttons, ShellAPI;

type

TForm11 = class(TForm)

DriveComboBox1: TDriveComboBox;

DirectoryListBox1: TDirectoryListBox;

FileListBox1: TFileListBox;

Button1: TButton;

Label1: TLabel;

SpeedButton1: TSpeedButton;

Label2: TLabel;

Label3: TLabel;

DriveComboBox2: TDriveComboBox;

Label4: TLabel;

BitBtn7: TBitBtn;

BitBtn2: TBitBtn;

procedure Button1Click(Sender: TObject);

procedure DriveComboBox1Change(Sender: TObject);

procedure DirectoryListBox1Change(Sender: TObject);

procedure BitBtn7Click(Sender: TObject);

procedure SpeedButton1Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

private

public

procedure Search;

procedure UnPack;

{ Public declarations }

end;

var

SerialNum: dword;

VolumeName, FSName : array [0..255] of char;

MaximumFNameLength,FileSystemFlags:dword;

CurCluster:DWORD;

//

Form11: TForm11;

s,s1:PWidechar;

hDevice:THandle;

buf1: array [0..108000] of byte;

bResult:LongBool;

lpBytesReturned:DWORD;

FS: TFileStream;

BS,attr,atrlenght:array [0..7] of byte;

DATA:array [0..512] of byte;

I,j,attrhead:byte;

bootsect,mftbyte:int64;

count,num,offset,t,first:integer;

atroffset,atrhead,lenght:integer;

select: array [0..20] of String;

cloffs,frstcl,lstcl:array[0..10] of int64; // смещение.первый кластер.последний кластер

mfcloffs,mffrstcl,mflstcl:array[0..10] of int64; // перенос для хранения значений отрезков DATA mft

directoryf,CurrentName:string[100];

FileName:string[50];

DirectoryNames:array [1..20] of string[50];

sr: TSearchRec;

implementation

uses Unit12, ABOUT;

{$R \*.dfm}

procedure TForm11.BitBtn2Click(Sender: TObject);

begin

form11.Close;

end;

procedure TForm11.BitBtn7Click(Sender: TObject);

begin

ShellExecute(0, 'open', Pchar(ExtractFilePath(Application.ExeName) + 'help.chm'), nil, nil, SW\_MAXIMIZE);

end;

procedure TForm11.Button1Click(Sender: TObject);

begin

s:=Pchar('\\.\'+form11.DriveCombobox1.Drive+':');

bResult:=GetVolumeInformation(s, VolumeName, SizeOf(VolumeName),

@SerialNum,

MaximumFNameLength,

FileSystemFlags,

FSName, SizeOf(FSName)) ;

//читаем Boot-сектор диска-оригинала

hDevice:=CreateFile(s, GENERIC\_READ, FILE\_SHARE\_READ or FILE\_SHARE\_WRITE,nil,OPEN\_EXISTING,0,0);

if hDevice=INVALID\_HANDLE\_VALUE then begin ShowMessage(IntToStr(GetLastError)); exit; end;

FillChar(buf1,sizeof(buf1),0);

if(hDevice<>INVALID\_HANDLE\_VALUE) then begin

SetFilePointer(hDevice,0,nil,FILE\_BEGIN);

bResult:=ReadFile(hDevice,buf1,512,lpBytesReturned,nil);

end;

j:=0; //BootSector

for I := 48 to 55 do

begin

BS[j]:=buf1[I];

inc(j);

end;

j:=0;

bootsect:=int64(BS); //первый кластер mft

mftbyte:=bootsect\*8\*512;

// считаем атрибут data из mft значение его 80h

//считаем всю mft

FillChar(buf1,sizeof(buf1),0);

if(hDevice<>INVALID\_HANDLE\_VALUE) then begin

SetFilePointerEx(hDevice,mftbyte,nil,FILE\_BEGIN);

bResult:=ReadFile(hDevice,buf1,1024,lpBytesReturned,nil); //читаем mft

end;

attrhead:=$80;

Search(); //Загружаем в массив DATA содержимое атрибута DATA

//раскрываем список отрезков DATA

offset:=64; // список отрезков начинается по смещению в атрибуте DATA с позиции 40h

UnPack(); // получаем список отрезков атрибута DATA таблицы mft

//cloffs,frstcl,lstcl-кластерное смещение,первыйкластер,последний кластер

for I := 0 to 10 do

begin

mfcloffs[I]:= cloffs[I];

mffrstcl[I]:= frstcl[I];

mflstcl[I]:= lstcl[I];

end;

if AnsiCompareText(FSName,'FAT32')=0

then begin

ShowMessage(S);

ShowMessage('Файловая система FAT32 не поддерживается');

exit;

end;

j:=1;

for I := 0 to 20 do

select[i]:='';

num:=0;

CurrentName:=ExtractFileName(DirectoryListBox1.Directory); //имя открытого каталога

FileName:=ExtractFileDir(ExpandUNCFileName(CurrentName));

first:=4;

for I := 1 to length(FileName) do

begin

if FileName[first+I]='\' then

begin

DirectoryNames[j]:=copy(FileName,first,I);

first:=first+I+1;

inc(j);

end;

end;

j:=0;

for I := 0 to (FileListBox1.Items.Count-1) do

begin

if FileListBox1.Selected[i] then

begin

select[num]:=ExtractFileName(FileListBox1.Items[i]); //имена файлов в массиве

num:=num+1;

end;

end;

t:=FileListBox1.Items.Count;

if FindFirst('C:\'+CurrentName +'\'+ '\*.\*', faAnyFile, sr) = 0 then

begin

repeat

if (sr.Attr and faDirectory) <> 0 then // если найденный файл - папка

begin

if (sr.Name <> '.') and (sr.Name <> '..') then // игнорировать служебные папки

begin

filename:= sr.Name;

end;

end;

until FindNext(sr) <> 0;

end;

FindClose(sr);

if select[0]='' then

begin

ShowMessage('Не выбран ни один файл');

exit;

end;

Form12.Show ;

end;

procedure TForm11.DirectoryListBox1Change(Sender: TObject);

begin

FileListBox1.Directory := DirectoryListBox1.Directory;

end;

procedure TForm11.DriveComboBox1Change(Sender: TObject);

begin

DirectoryListBox1.Drive := DriveComboBox1.Drive;

end;

procedure TForm11.Search; //процедура поиска атрибута DATA и его считывание в массив

var

I,j:integer;

begin

j:=0; //смещение первого атрибута

for I := 20 to 21 do

begin

attr[j]:=buf1[I];

inc(j);

end;

j:=0;

atroffset:=int64(attr);

while buf1[atroffset]<>attrhead do

begin

j:=0;

for I := atroffset+4 to atroffset+7 do // если заголовок не соответствует 80h то,вычисляем длину атр

begin

atrlenght[j]:=buf1[I];

inc(j);

end;

atroffset:=int64(atrlenght)+atroffset;

end;

j:=0;

for I := atroffset+4 to atroffset+7 do

begin

atrlenght[j]:=buf1[I];

inc(j);

end;

lenght:=int64(atrlenght);

j:=0;

for I :=atroffset to atroffset+ lenght do //считываем все DATA

begin

DATA[j]:=buf1[I];

inc(j);

end;

end;

procedure TForm11.SpeedButton1Click(Sender: TObject);

begin

AboutBox.ShowModal;

end;

procedure TForm11.UnPack;

var

lo,hi,analyz:byte;

firstcluster,clusteroffset,lastcluster: array [0..7] of byte;

I1,I,j:byte;

begin

I1:=0;

j:=0;

FillChar(firstcluster,sizeof(firstcluster),0);

FillChar(clusteroffset,sizeof(clusteroffset),0);

FillChar(lastcluster,sizeof(lastcluster),0);

while DATA[offset]<>0 do

begin

analyz:=DATA[offset];

hi:=(analyz and $F0)shr 4; //размер начального кластера (старшие 4 разряда)

lo:=analyz and $0F; //размер смещения (младшие 4 разряда)

for I := offset+1 to offset+lo do

begin

clusteroffset[j]:=DATA[I];

inc(j);

end;

j:=0;

cloffs[I1]:=int64(clusteroffset);

for I := offset+1+lo to offset+lo+hi do

begin

firstcluster[j]:=DATA[I];

inc(j);

end;

j:=0;

frstcl[I1]:=int64(firstcluster);

if I1>0 then

frstcl[I1]:= frstcl[I1]+ frstcl[I1-1];

lstcl[I1]:=frstcl[I1]+cloffs[I1];

offset:=offset+lo+hi+1;

inc(I1);

end;

end;

end.

## **7.2. Исходный текст модуля Unit12 (файл Unit12.pas)**

unit Unit12;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.FileCtrl, Vcl.ComCtrls,Unit11,

Vcl.Buttons,ShellAPI;

type

TForm12 = class(TForm)

Memo1: TMemo;

Memo2: TMemo;

Label3: TLabel;

Label4: TLabel;

SpeedButton3: TSpeedButton;

SpeedButton4: TSpeedButton;

BitBtn2: TBitBtn;

SpeedButton1: TSpeedButton;

BitBtn3: TBitBtn;

BitBtn4: TBitBtn;

SpeedButton2: TSpeedButton;

SaveDialog1: TSaveDialog;

BitBtn1: TBitBtn;

BitBtn5: TBitBtn;

BitBtn7: TBitBtn;

procedure SpeedButton3Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure SpeedButton4Click(Sender: TObject);

procedure SpeedButton1Click(Sender: TObject);

procedure SpeedButton2Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure BitBtn4Click(Sender: TObject);

procedure BitBtn3Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn5Click(Sender: TObject);

procedure BitBtn7Click(Sender: TObject);

private

{ Private declarations }

public

procedure iaSearchFile;

procedure iaSearchDir;

procedure GetFileRecordDir;

procedure GetFileRecordFile;

end;

var

Form12: TForm12;

diradr,fladr:int64;

I1,z,I2,m:byte;

dirnum,currentdirnum:integer; //номер файловой записи каталога

flnum:array [0..20] of integer; // номера файловых записей файлов

dir :array [1..4] of word;

filedir :array [1..4] of word;

wideChars : array[0..99] of WideChar;

i:word;

bytes:integer;

flcloffs,flfrstcl,fllstcl:array[0..20,0..10] of int64;

names: string;

directory:string;

Thread1,Thread2:TThread;

implementation

{$R \*.dfm}

uses Unit4,Unit5;

procedure TForm12.SpeedButton1Click(Sender: TObject);

begin

Savedialog1.FileName:='Copy.txt';

if SaveDialog1.Execute then

Memo1.Lines.SaveToFile(Savedialog1.FileName);

end;

procedure TForm12.SpeedButton2Click(Sender: TObject);

begin

Savedialog1.FileName:='Analis.txt';

if SaveDialog1.Execute then

Memo2.Lines.SaveToFile(Savedialog1.FileName);

end;

procedure TForm12.SpeedButton3Click(Sender: TObject);

begin

//переход к корневому каталогу,он всегда 5ая файловая запись

dirnum:=5;

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

j:=1;

while DirectoryNames[j]<>'' do

begin

directoryf:=DirectoryNames[j];

GetFileRecordDir();

inc(j);

end;

directoryf:=CurrentName;

GetFileRecordDir(); //получаем номер файловой записи каталога

currentdirnum:=dirnum;

GetFileRecordFile(); //получим номера файловых запесей выбранных файлов внутри каталога

Thread2:=ThrAnalys.Create(false);

Thread2.Priority:=tpLowest;

SpeedButton3.Enabled:=False;

end;

procedure TForm12.SpeedButton4Click(Sender: TObject);

begin

directory:=CurrentName;

Thread1:=ThrCopy.Create(false);

Thread1.Priority:=tpLowest;

SpeedButton4.Enabled:=False;

end;

procedure TForm12.BitBtn1Click(Sender: TObject);

begin

if Thread2.Suspended= true then Thread2.Resume;

Thread2.Destroy;

BitBtn5.Enabled:=False;

BitBtn1.Enabled:=False;

form12.Memo2.Lines.Add('Анализ прерван пользователем');

end;

procedure TForm12.BitBtn2Click(Sender: TObject);

begin

form12.Close;

end;

procedure TForm12.BitBtn3Click(Sender: TObject);

begin

if Thread1.Suspended= true then Thread1.Resume;

Thread1.Destroy;

BitBtn4.Enabled:=False;

BitBtn3.Enabled:=False;

form12.Memo1.Lines.Add('Копирование прервано пользователем');

end;

procedure TForm12.BitBtn4Click(Sender: TObject);

begin

if Thread1.Suspended=false then Thread1.Suspend

else thread1.Resume;

end;

procedure TForm12.BitBtn5Click(Sender: TObject);

begin

if Thread2.Suspended=false then Thread1.Suspend

else thread2.Resume;

end;

procedure TForm12.BitBtn7Click(Sender: TObject);

begin

ShellExecute(0, 'open', Pchar(ExtractFilePath(Application.ExeName) + 'help.chm'), nil, nil, SW\_MAXIMIZE);

end;

procedure TForm12.FormCreate(Sender: TObject);

begin

memo1.Lines.Clear;

memo2.Lines.Clear;

end;

procedure TForm12.GetFileRecordDir;

begin

diradr:=(mffrstcl[0]\*8\*512)+dirnum\*2\*512;

FillChar(buf1,sizeof(buf1),0);

if(hDevice<>INVALID\_HANDLE\_VALUE) then begin

SetFilePointerEx(hDevice,diradr,nil,FILE\_BEGIN);

bResult:=ReadFile(hDevice,buf1,1024,lpBytesReturned,nil); //читаем корневой каталог

end;

attrhead:=$A0;

FillChar(DATA,sizeof(DATA),0);

FillChar(attr,sizeof(attr),0);

FillChar(atrlenght,sizeof(atrlenght),0);

atroffset:=0;

// в DATA будет лежать аттрибут allocation index для корневого каталога

Form11.Search();

// теперь надо найти отрезки .длина атрибута IA равна 4,значит отрезки начинаются со смещения 48h

offset:=72; //48h

FillChar(cloffs,sizeof(cloffs),0);

FillChar(frstcl,sizeof(frstcl),0);

FillChar(lstcl,sizeof(lstcl),0);

Form11.UnPack(); // получаем список отрезков атрибута IA

//cloffs,frstcl,lstcl-кластерное смещение,первыйкластер,последний кластер

z:=0;

FillChar(wideChars,sizeof(wideChars),0);

StringToWideChar(directoryf, wideChars, 100); //юникод русских символов

FillChar(dir,sizeof(dir),0);

for I1 := 1 to 4 do //помещаем имя каталога в массив

begin

dir[I1]:=ord(wideChars[I1-1]);

end;

while cloffs[z]<> 0 do

begin

for m := 0 to cloffs[z]-1 do

begin

bytes:=8\*512; //кластер,т.к в каждом кластере новая индексная запись

diradr:=(frstcl[z]\*8\*512)+bytes\*m;

FillChar(buf1,sizeof(buf1),0);

if(hDevice<>INVALID\_HANDLE\_VALUE) then begin

SetFilePointerEx(hDevice,diradr,nil,FILE\_BEGIN);

bResult:=ReadFile(hDevice,buf1,bytes,lpBytesReturned,nil); //читаем index allocation корневого каталога

end;

iaSearchDir; // ищем номер файловой записи выбранного каталога

end;

inc(z);

end;

end;

procedure TForm12.GetFileRecordFile;

begin

diradr:=(mffrstcl[0]\*8\*512)+dirnum\*2\*512;

FillChar(buf1,sizeof(buf1),0);

if(hDevice<>INVALID\_HANDLE\_VALUE) then begin

SetFilePointerEx(hDevice,diradr,nil,FILE\_BEGIN);

bResult:=ReadFile(hDevice,buf1,1024,lpBytesReturned,nil); //читаем выбранный каталог

end;

attrhead:=$A0;

FillChar(DATA,sizeof(DATA),0);

FillChar(attr,sizeof(attr),0);

FillChar(atrlenght,sizeof(atrlenght),0);

atroffset:=0;

// в DATA будет лежать аттрибут allocation index для выбранного каталога

Form11.Search();

// теперь надо найти отрезки .длина атрибута IA равна 4,значит отрезки начинаются со смещения 48h

offset:=72; //48h

FillChar(cloffs,sizeof(cloffs),0);

FillChar(frstcl,sizeof(frstcl),0);

FillChar(lstcl,sizeof(lstcl),0);

Form11.UnPack(); // получаем список отрезков атрибута IA

//cloffs,frstcl,lstcl-кластерное смещение,первыйкластер,последний кластер

for I2 := 0 to num-1 do

begin

StringReplace(select[I2], #13, '', [rfReplaceAll]);

StringReplace(select[I2], #10, '', [rfReplaceAll]);

z:=0;

FillChar(wideChars,sizeof(wideChars),0);

StringToWideChar( select[I2], wideChars, 100); //юникод русских символов

FillChar(filedir ,sizeof(dir),0);

for I1 := 1 to 4 do //помещаем имя файла в массив

begin

filedir [I1]:=ord(wideChars[I1-1]);

end;

while cloffs[z]<> 0 do

begin

for m := 0 to cloffs[z]-1 do

begin

bytes:=8\*512; //кластер,т.к в каждом кластере новая индексная запись

diradr:=(frstcl[z]\*8\*512)+bytes\*m;

FillChar(buf1,sizeof(buf1),0);

if(hDevice<>INVALID\_HANDLE\_VALUE) then begin

SetFilePointerEx(hDevice,diradr,nil,FILE\_BEGIN);

bResult:=ReadFile(hDevice,buf1,bytes,lpBytesReturned,nil); //читаем index allocation

end;

iaSearchFile(); // ищем номер файловой записи выбранного файла

end;

inc(z);

end;

end;

end;

procedure TForm12.iaSearchDir;//для каталога

var

iabuf: array [0..7] of byte;

name: array [1..4] of word;

I:longint;

j:byte;

ravno:bool;

frstentryindex,indsize,nameoffset,namedir:longint;

a:integer;

begin

FillChar(name,sizeof(name),0);

FillChar(iabuf,sizeof(iabuf),0);

j:=0;

for I := 24 to 27 do

begin

iabuf[j]:=buf1[I];

inc(j);

end;

j:=0;

frstentryindex:=int64(iabuf)+24; //начало индексной записи относительно ia

FillChar(iabuf,sizeof(iabuf),0);

ravno:=false;

while ravno=false do

begin

FillChar(name,sizeof(name),0);

FillChar(iabuf,sizeof(iabuf),0);

j:=0;

for I := frstentryindex+8 to frstentryindex+9 do

begin

iabuf[j]:=buf1[I];

inc(j);

end;

j:=0;

indsize:=int64(iabuf); //размер индексной записи

FillChar(iabuf,sizeof(iabuf),0);

for I := frstentryindex+10 to frstentryindex+11 do

begin

iabuf[j]:=buf1[I];

inc(j);

end;

j:=0;

nameoffset:=int64(iabuf); //начало file name

FillChar(iabuf,sizeof(iabuf),0);

//один символ 2 байта ,берем для сравнения первых 4 символа ,читаем 8 байт

a:=82-nameoffset;

j:=1;

for I := frstentryindex+nameoffset+a to frstentryindex+nameoffset+a+8 do

begin

if (I mod 2)<>0 then

begin

name[j]:=name[j]+(buf1[I] shl 8);

inc(j);

end;

if (I mod 2)=0 then

begin

name[j]:=name[j]+buf1[I];

end;

end;

j:=0;

for I := frstentryindex to frstentryindex+5 do

begin

iabuf[j]:=buf1[I];

inc(j);

end;

j:=0;

dirnum:=int64(iabuf); //носер фс

FillChar(iabuf,sizeof(iabuf),0);

frstentryindex:=frstentryindex+indsize;

if name[1]=dir[1] then

if name[2]=dir[2] then

if name[3]=dir[3] then

if name[4]=dir[4] then

ravno:=True;

if indsize = 0 then // если кол-во кластеров н-го прочитано и каталог не найден выйти из процедуры и читать след отр

ravno:=True;

end;

end;

procedure TForm12.iaSearchFile; //для файлов

var

iabuf: array [0..7] of byte;

name: array [1..4] of word;

I:longint;

j:byte;

ravno:bool;

frstentryindex,indsize,nameoffset:longint;

a:integer;

begin

FillChar(name,sizeof(name),0);

FillChar(iabuf,sizeof(iabuf),0);

j:=0;

for I := 24 to 27 do

begin

iabuf[j]:=buf1[I];

inc(j);

end;

j:=0;

frstentryindex:=int64(iabuf)+24; //начало индексной записи относительно ia

FillChar(iabuf,sizeof(iabuf),0);

ravno:=false;

while ravno=false do

begin

FillChar(name,sizeof(name),0);

FillChar(iabuf,sizeof(iabuf),0);

j:=0;

for I := frstentryindex+8 to frstentryindex+9 do

begin

iabuf[j]:=buf1[I];

inc(j);

end;

j:=0;

indsize:=int64(iabuf); //размер индексной записи

FillChar(iabuf,sizeof(iabuf),0);

for I := frstentryindex+10 to frstentryindex+11 do

begin

iabuf[j]:=buf1[I];

inc(j);

end;

j:=0;

nameoffset:=int64(iabuf); //начало file name

FillChar(iabuf,sizeof(iabuf),0);

//один символ 2 байта ,берем для сравнения первых 4 символа ,читаем 8 байт

a:=82-nameoffset; //отриц чисел не может быть,меняем местами

j:=1;

for I := frstentryindex+nameoffset+a to frstentryindex+nameoffset+a+8 do

begin

if (I mod 2)<>0 then

begin

name[j]:=name[j]+(buf1[I] shl 8);

inc(j);

end;

if (I mod 2)=0 then

begin

name[j]:=name[j]+buf1[I];

end;

end;

j:=0;

for I := frstentryindex to frstentryindex+5 do

begin

iabuf[j]:=buf1[I];

inc(j);

end;

j:=0;

flnum[I2]:=int64(iabuf); //носер фс

FillChar(iabuf,sizeof(iabuf),0);

frstentryindex:=frstentryindex+indsize;

if name[1]=filedir[1] then

if name[2]=filedir[2] then

if name[3]=filedir[3] then

if name[4]=filedir[4] then

ravno:=True;

if indsize = 0 then // если кол-во кластеров н-го прочитано и каталог не найден выйти из процедуры и читать след отр

ravno:=True;

end;

end;

end.

## **7.3. Исходный текст модуля Unit4 (файл Unit4.pas)**

unit Unit4;

interface

uses

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

Dialogs, StdCtrls, FileCtrl,ComCtrls, Buttons;

type

ThrCopy = class(TThread)

private

ExitProcFlag:boolean;

procedure CopySector;

procedure DirSearch(Dir: string);

procedure CopySectbysect;

protected

procedure Execute; override;

destructor Destroy; override;

end;

implementation

Uses Unit12,Unit11;

procedure ThrCopy.CopySectbysect;

begin

s:=Pchar('\\.\'+form11.DriveCombobox1.Drive+':');

s1:=Pchar('\\.\'+form11.DriveCombobox2.Drive+':');

SetCurrentDir(s1+'\');

CreateDir(s1+'\'+CurrentName);

DirSearch(FileName);

if ExitProcFlag then exit;

ShowMessage('Копирование каталога "'+directory+'" и его содержимого успешно завершено!');

end;

procedure ThrCopy.DirSearch(Dir: string);

var

SearchRec: TSearchRec;

j:byte;

begin

Dir := IncludeTrailingBackslash(Dir);

j:=0;

if FindFirst(Dir + '\*.\*', faAnyFile, SearchRec) = 0 then

repeat

Application.ProcessMessages;

if (SearchRec.Attr and faDirectory) <> 0 then begin

if (SearchRec.name <> '.') and (SearchRec.name <> '..') then

begin

CreateDir(s1+'\'+copy(Dir,3,length(Dir)-7)+'\'+SearchRec.Name);

DirSearch(Dir + SearchRec.name);

end;

end

else

begin

if ExitProcFlag then exit;

CurrentName:=Copy(Dir,4,length(Dir)-3);

names:=ExtractFileName(SearchRec.Name);

CopySector;

names:='';

end;

until FindNext(SearchRec) <> 0;

FindClose(SearchRec);

end;

procedure ThrCopy.CopySector; //копирование по секторам

var

iFileHandle,iFileHandle1: Integer;

iFileLength: Integer;

iBytesRead: Integer;

Buffer: PChar;

i,j: Integer;

m:integer;

begin

I:=0;

Form12.Memo1.Lines.Add('Копирование файла:'+names);

SetCurrentDir(FileName);

iFileHandle1:=CreateFile(Pchar(Names), GENERIC\_READ, FILE\_SHARE\_READ or FILE\_SHARE\_WRITE,nil,OPEN\_ALWAYS,0,0);

FileClose(iFileHandle1);

iFileHandle := FileOpen(FileName+'\'+Names, fmOpenRead);

iFileLength := FileSeek(iFileHandle,0,2);

iFileHandle1 := FileOpen(s1+'\'+CurrentName+'\'+Names, fmOpenWrite);

Buffer := PChar(AllocMem(512 + 1)); //копирование по блокам равным одному сектору

while I<=iFileLength do

begin

if ExitProcFlag then exit;

FileSeek(iFileHandle,0,I);

FileSeek(iFileHandle1,0,I);

iBytesRead := FileRead(iFileHandle, Buffer^, 512);

iBytesRead:=FileWrite(iFileHandle1, Buffer^, 512);

m:=I div 512;

Form12.Memo1.Lines.Add('Скопирован сектор:'+inttostr(m));

I:=I+512;

end;

FileClose(iFileHandle);

FileClose(iFileHandle1);

FreeMem(Buffer);

inc(j);

end;

procedure ThrCopy.Execute;

begin

CopySectbysect;

{ Place thread code here }

end;

destructor ThrCopy.Destroy;

begin

ExitProcFlag:=true;

inherited Destroy;

end;

end.

## **7.4. Исходный текст модуля Unit5 (файл Unit5.pas)**

unit Unit5;

interface

uses

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

Dialogs, StdCtrls, FileCtrl,ComCtrls, Buttons;

type

ThrAnalys = class(TThread)

private

ExitProcFlag:boolean;

procedure AnalysCluster;

protected

procedure Execute; override;

destructor Destroy; override;

end;

implementation

Uses Unit12,Unit11;

procedure ThrAnalys.AnalysCluster;

var

I:integer;

begin

for I := 0 to num-1 do

begin

if ExitProcFlag then exit;

Form12.Memo2.Lines.Add('Последовательность отрезков для файла:'+select[I]);

fladr:=(mffrstcl[0]\*8\*512)+flnum[I]\*2\*512;

FillChar(buf1,sizeof(buf1),0);

if(hDevice<>INVALID\_HANDLE\_VALUE) then begin

SetFilePointerEx(hDevice,fladr,nil,FILE\_BEGIN);

bResult:=ReadFile(hDevice,buf1,1024,lpBytesReturned,nil); //читаем атрибут файла

end;

attrhead:=$80;

Form11.Search(); //Загружаем в массив DATA содержимое атрибута DATA

//раскрываем список отрезков DATA

offset:=64; // список отрезков начинается по смещению в атрибуте DATA с позиции 40h

Form11.UnPack(); // получаем список отрезков атрибута DATA файла

//cloffs,frstcl,lstcl-кластерное смещение,первыйкластер,последний кластер

z:=0;

while cloffs[z]<>0 do

begin

flcloffs[I][z]:=cloffs[z];

flfrstcl[I][z]:=frstcl[z];

fllstcl[I][z]:=lstcl[z];

Form12.Memo2.Lines.Add(inttostr(z)+' отрезок = Начальный кластер('+inttostr(frstcl[z])+'); '+' Последний кластер('+inttostr(lstcl[z])+');');

inc(z);

end;

end;

ShowMessage('Анализ для '+inttostr(num)+' файлов успешно завершен!');

end;

procedure ThrAnalys.Execute;

begin

AnalysCluster;

end;

destructor ThrAnalys.Destroy;

begin

ExitProcFlag:=true;

inherited Destroy;

end;

end.

# 8. Список литературы

1. Методические указания к лабораторным работам по курсу «Операционные системы»/РГРТА Сост. Засорин С.В. Рязань, 2002, 56 с.
2. Архангельский А. Я. «Delphi 7. Справочное пособие»/Бином, Москва, 2003, 1024с.
3. Баженова И. Ю. «Delphi 7. Самоучитель программиста»/М, Кудиц-образ, 2003, 448с.
4. Шамис В. А. «Borland C++ Builder 6. Для профессионалов./СПб, Питер, 2004, 798с.
5. MSDN Library – May 2004.