Министерство образования и науки Российской Федерации

Государственное образовательное учреждение высшего профессионального образования

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ЭВМ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по дисциплине “Операционные системы”

по теме “Разработка вспомогательной системной программы в системе программирования Delphi с использованием средств WinApi”

Выполнил студент группы 742

Шипилов Д.А.

Руководитель Доц. Каф. ЭВМ

Засорин С.В.

Рязань 2010

# Содержание

Введение

1. Техническое задание

1.1 Основания для разработки программы

1.2 Назначение разработки

1.3 Требования к программам

1.4 Требования к надежности

1.5 Требования к программной документации

1.6 Текст задания по варианту

2. Описания разработанных программ

2.1 Общие сведения

2.2 Функциональное назначение

2.3 Описание логической структуры

2.3.1 Логическая структура программы

2.3.2 Логическая структура справочной системы

2.3.3 Создание программы-инсталлятора

2.4 Используемые технические средства

2.5 Вызов и загрузка

2.6 Входные данные

2.7 Выходные данные

3. Программа и методика испытаний

3.1 Объект испытаний

3.2 Цель испытаний

3.3 Требования к программе

3.4 Требования к программной документации

3.5 Средства и порядок испытаний

3.6 Методы испытаний

3.6.1 Тестирование программы методом белого ящика

3.6.2 Тестирование программы методом чёрного ящика

4. Эксплуатационные документы

4.1 Руководство программиста

4.1.1 Общие сведения о программе

4.1.2 Структура программ

4.1.3 Проверка программ

4.1.4 Дополнительные возможности

4.1.5 Сообщения системному программисту

4.2. Руководство оператора (пользователя)

4.2.1 Назначение и условия применения программ

4.2.2 Обращение к программам для запуска

4.2.3 Входные и выходные данные

4.2.4 Сообщения оператору

Приложение

Список литературы

**Введение**

В данной курсовой работе было разработано приложение осуществляющее передачу данных между копиями этого приложения с помощью потоков, а также анализ системной области рабочего. Анализ производится путём ассоциирования жёсткого диска с файлом программы и дальнейшей работы с дескриптором этого файла.

1**. Техническое задание**

## 1.1 Основания для разработки программы

Основанием для разработки программы являлось задание для курсового проекта (вариант 21).

Содержание задания:

Разработать многопоточное приложение, выполняющее различные операции:

1)Обмен данными между двумя процессами с помощью файла проецируемого в память.

2)Анализ содержимого служебной области системного диска.

## 

## 1.2 Назначение разработки

Программа служит для обмена текстовыми сообщениями между копиями этой программы, а также для анализа активных носителей, присутствующих в системе. Конкретно, происходит анализ блока параметров BIOS выбранного логического диска на носителе, и считывание и анализ таблицы разделов этого носителя.

## 

## 1.3 Требования к программам

Приложение должно удовлетворять следующим требованиям:

1. Обеспечивать указание данных для обмена в одном из процессов
2. Выполнять чтение и анализ содержимого служебной области, расшифровку её содержимого и представление результатов в виде таблицы на устройство стандартного вывода или в файле.

В данном приложении реализован способ представления результатов работы в устройстве стандартного вывода информации - мониторе.

Для удобства модификации приложения оно выполнено в виде нескольких функциональных модулей. Первая группа модулей служит для обмена данными между различными копиями приложения. Вторая группа модулей служит для считывания рабочей области системного, или какого-либо другого носителя информации. Приложение позволяет получить информацию о таблице разделов физического носителя и блоке параметров BIOS этого носителя.

## 1.4 Требования к надежности

Программа должна выдавать сообщения об ошибках в следующих случаях:

1)Если не удалось создать проекцию файла

2)Не удалось синхронизировать приложение с другими копиями приложения

3)Не удалось осуществить обмен данными

4) Если не введено имя при регистрации пользователя в системе

## 1.5 Требования к программной документации

Приложение должно содержать справочную систему , включающую в себя всю необходимую информацию по работе программы.

## 1.6 Текст задания по варианту

Разработать многопоточное приложение, выполняющее операции.

1)Обмен данными между двумя процессами с помощью файлов проецируемых в память.

2)Анализ содержимого служебной области системного диска.

**Исходные данные:**

Для выполнения первой операции использовать прикладные программы типа простых текстовых редакторов или графических редакторов.

Имя системного диска.

Полное имя файла для вывода результатов.

**Технические требования и условия:**

Приложение должно выполнять следующие операции:

Обеспечивать возможность указывать данные для обмена в одном из процессов.

Выполнять чтение и анализ служебной области , расшифровку её содержимого и представление результатов в виде таблицы на устройстве стандартного вывода или в файле.

**2. Описания разработанных программ**

## 2.1 Общие сведения

Для выполнения программы необходима ОS Windows 9x/2000/XP, а также более поздние модели такие как Windovs Vista или Windows7/

Программы написаны на языке программирования Object Pascal в среде разработки Delphi 7.

## 2.2 Функциональное назначение

Программа предназначена для обмена данными между пользователями, работающими на одной вычислительной системе под ос Windows на разных терминалах. Также программа осуществляет анализ служебной области системного диска (Partition Table носителя и блока параметров Bios логического диска того же носителя )

**2.3 Описание логической структуры**

**2.3.1 Логическая структура программы**

*Этап1*

Работа программы начинается с запуска приложения OffClient.exe. Перед переходом на главную форму программы появляется окно регистрации (рисунок 1.).

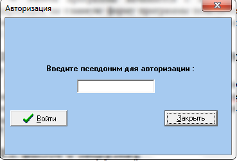


Рисунок 1. Окно регистрации в программе.

На этом этапе возможно наступление следующих событий :

1) «Событие1» – Кнопка «Войти»;

2) «Событие2» – Кнопка «Закрыть»;

Алгоритмы обработчиков событий этой формы.

Глобальные переменные:

Form2: TForm2 – переменная класса формы, отвечающая за её состояние

formClose:boolean – булевская переменная отвечающая за проверку разрешения закрытия формы.

Также есть возможность обращаться к объекту TForm1 через подключение к модулю этой формы модуля главной формы (в рабочем каталоге приложения главный модуль имеет имя «Unit1»).

1. Описание события «Событие1»:

Выполнение обработчика начинается при нажатии кнопки «Войти».

Для успешного выполнения обработчика события «Событие1» необходимо ввести псевдоним в соответствующее поле. После нажатия на кнопку «войти» происходит проверка поля ввода псевдонима, если поле пустое , то выводится сообщение о том, что необходимо ввести псевдоним, если поле псевдонима не пустое, то специальной переменной булевского типа, которая сигнализирует о том, что форма регистрации может быть закрыта присваивается значение TRUE. После этого, происходить передача псевдонима в основную форму программы. Очистка поля Memo1 на главной форме программы. Далее происходит вызов процедуры из главного модуля программы WriteDatainMMF для оповещения других пользователей о том что подключился новый пользователь. Далее происходит закрытие данной формы и переход к основной форме программы.

Алгоритм обработки данного события представлен на схеме – (блок схема1)



Блок схема 1 Алгоритм обработки события «Событие 1»

Данный алгоритм реализует процедуру регистрации пользователя в системе под определённым именем «NICKNAME», которое он вводит в поле псевдонима. При этом программа оповещает других пользователей, которые уже зарегистрированы, о подключении ещё одного пользователя с именем «NICKNAME». После регистрации каждое отправленное сообщение будет маркировано как сообщение от определённого пользователя с именем «NICKNAME».

1. Описание события «Событие2»:

Выполнение обработчика события начинается при нажатии кнопки «Закрыть».

Алгоритм выполнения обработчика представлен на схеме «блок схема 2».



## Этап 2

На этом этапе происходит переход к главной форме приложения(рисунок2). и происходит обмен сообщениями помощью файла , проецируемого в память. Также на этой форме имеются кнопки для перехода к окну информации о системных носителях. Это сделано исходя из задания на курсовой проект

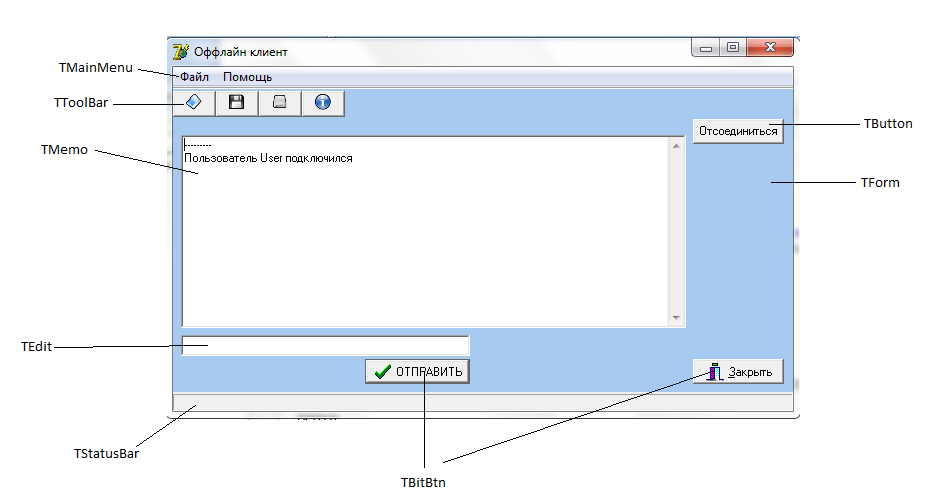


Рисунок2. Форма основного приложения.(программно Form2).

На этом этапе возможны следующие события:

1. «Событие1» при на жатии на кнопку «ОТПРАВИТЬ»
2. «Событие2»при нажатии кнопку «Отсоединиться»
3. «Событие3» при нажатии на кнопку «Соединиться»
4. «Событие4» при нажатии на кнопку с изображение каталога.
5. «Событие5» при нажатии на кнопку с изображением дискеты
6. «Событие6» при нажатии на кнопку с изображением восклицательного занка.
7. «Событие7» при нажатии на кнопку «Закрыть»
8. «Событие8» при нажатии на кнопку с изображением жёсткого диска.

Алгоритмы обработчиков событий для основной формы программы:

Глобальные переменные:

TxtFile:TextFile; Файл для сохранения и загрузки истории переписки

St:PansiChar; .// строка для хранения текста сообщений об ошибках и программных сообщениях.

lincInt:Boolean; переменная для проврки правильности соединения

TxtContaningStr:String;// строка , содержащая текст. Используется для хранения текста загружаемой и сохраняемой ситории сообщений.

userName:string; Переменная хранящая имя пользователя, работающего с программой.

Также в программе используются публичные переменные , доступные во всей сборке(всех модулях данного приложения):

HMutex:THandle; - дескриптор объекта типа

ThreadRec:TThreadReceiver;-дескриптор объекта типа поток, для принятия сообщений посланных из первичного потока приложения.

HEvent:THandle;- дескриптор объекта типа событие. Необходим для сообщения вторичным потокам об отправке данных из первичного потока

HFileSender:Thandle;-Дескриптор файла , проецируемого в память, необходим для обращения к файлу ,проецируемому в память, и обмена через него информацией с другими приложениями.

1. Описание события «Событие1»:

Наступление события происходит при нажатии на кнопку «ОТПРАВИТЬ». После нажатия на кнопку «ОТПРАВИТЬ» происходит проверка присутствия данных в поле ввода сообщения , если данные отсутствуют, то происходит выход из процедуры, если данные присутствуют, то происходит вызов процедуры WriteDataInMMF. В данной процедуре происходит ожидания объекта типа Mutex в течении бесконечного интервала времени. Это необходимо для реализации обмена сообщениями между несколькими копиями приложения. В данный момент происходит попытка захвата разделяемого ресурса - файа, проецируемого в память. После захвата объекта типа MUTEX происходит проецирование файла на адресное пространство основного процесса. Если файл спроецирован успешно, то происходит запись данных в этот файл, которые становятся доступны для чтения другим процессам, а затем проекция удаляется. Если проекцию создать не удалось, то выводится сообщение: «Не удалось создать проекцию файла». Алгоритм представлен на схемеах(блок схема 2.1, блок схема 2.2).



Блок схема 2.1 Алгоритм обработки события «Событие1».



Блок схема 2.2 Алгоритм процедуры передачи данных.

Данный алгоритм реализуется при нажатии на кнопку «ОТПРАВИТЬ» . В теле обработчика события «Событие1» вызывается данная процедура.(передачи данных. Программно «WriteDataInMMF»)

1. Описание события «Событие2»:

Данное событие происходит при нажатии на кнопку «Отсоединиться». После нажатия на кнопку «Отсоединится» происходит вызов метода Terminate для потока, который осуществляет приём данный, то есть считывание их из проецируемого файла. Далее , пока поток не остановлен, В проецируемый файл, с помощью рассмотренной выше функции WriteDataInMMF Записываются данные о том, что пользователь отключился. Далее происходит ожидание завершения потока и освобождение ресурсов памяти выделенных этому потоку. После этого происходит уничтожение дескрипторов всех объектов необходимых для обмена информацией таких как: файл проецируемый в память , обьект типа mutex и объект типа, событие, который необходим, для оповещения вторичных потоков приложений о том, что данные записаны в проецируемый файл. Алгоритм работы представлен на схеме (блок схема 2.3).



Блок схема 2.3 Алгоритм обработчика события «Событие2».

1. Описание события «Событие3». Событие происходит при нажатии на кнопку «Соединится». При нажатии на данную кнопку происходит создание дескрипторов объектов , необходимых для обмена, эти дескрипторы были рассмотрены выше. Далее происходит запуск на выполнение вторичного потока приложения, который осуществляет считывание данных из проецируемого файла после их записи. Далее происходит запись данных в проецируемый файл с помощью функции. Алгоритм работы представлен на схеме(блок схема 2.4).



Блок схема алгоритма события «Событие3».

1. Описание события «Событие4».

Загрузка истории сообщений пользователей Событие наступает при нажатии на кнопку: . После нажатия на данную кнопку происходит запуск объекта Open Dialog, в котором выбирается файл для загрузки, далее происходит создание файловой переменной и ассоциирование её с выбранным в Open Dialog файлом, далее в цикле происходит считывание строк из файла и последовательная запись их в поле истории переписки. После считывания файла целиком цикла завершается и происходит закрытие файла. Алгоритм обработки данного события представлен на схеме (блок схема 2.5).



Блок схема 2.5 Обработчик события «Событие4».

1. Описание события «Событие 5». Сохранение истории переписки участников чата.

Событие наступает при нажатии на кнопку: . При нажатии на эту кнопку появляется диалоговое окно , где предлагается задать имя файлу и каталог, в котором этот файл будет сохранён, далее происходит ассоциирование созданного файла с файловой переменной и открытие этого файла для записи. Построчно текст из поля истории переписки заносится в строковую переменную , а затем в файл. Алгоритм обработчика события данной процедуры представлен на схеме(блок схема 2.6).



Блок схема 2.6. Алгоритм обработчика события «Событие 5».

1. Описание события «Событие 6». Это событие происходит по нажатию кнопки. При этом происходит вызов предметного указателя файла справки, который находится в рабочем каталоге приложения.



1. Описание события «Событие7». Происходит по нажатию на кнопку «Закрыть».Алгоритм обработчика представлен на схеме (блок схема 2.7).



Блок схема 2.7 обработчик события «Событие7».

1. Описание события «Событие8». Это событие происходит по нажатию на кнопку: . При этом происходит создание третьей формы приложения, переход на которую является третьим этапом построения приложения. Алгоритм обработчика события представлен на схеме (блок схема 2.8).



Блок схема 2.7.

*Этап3*

На этом этапе происходит переход к третьему окну приложения. На данном этапе реализуется вторая часть задания на курсовой проект – анализ служебной области системного диска: Форма3 приложения OffClient имеет заголовок «Служебная информация о системных носителях» её вид представлен на рисунке 3.

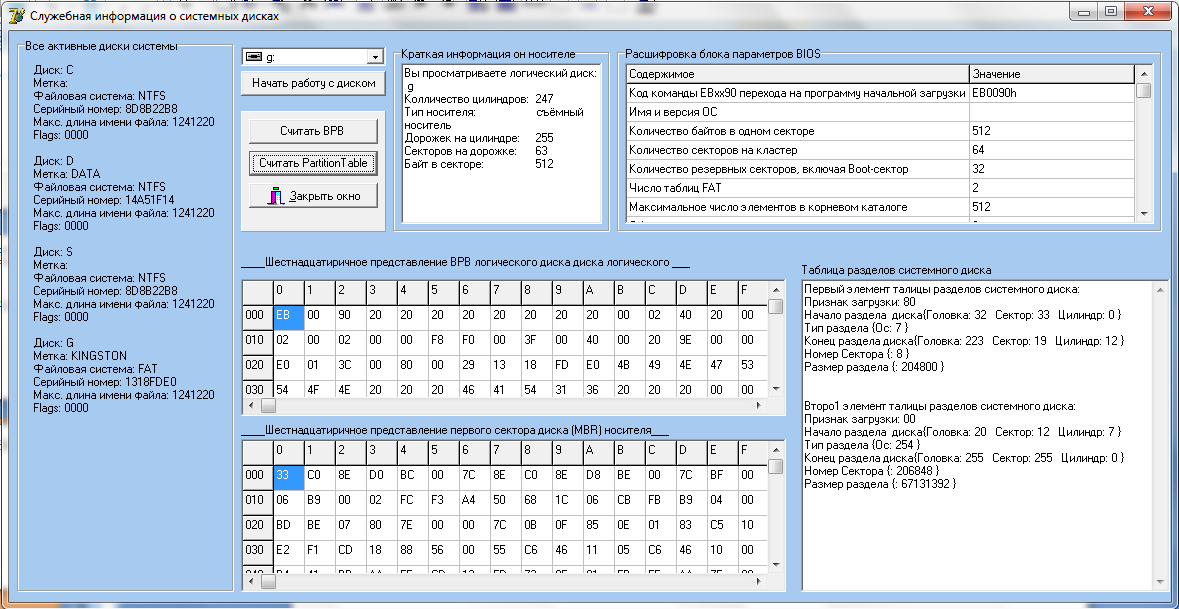


Рисунок 3. Окно отображения служебной информации о носителе.

При работе с данной формой возникает 4 события:

С1-начать работу с диском

С2-Считать BPB (Блок параметров BioS)

C3-Счтать Partition Table.(Таблицу разделов диска).

С4-Закрытие формы. Данное событие имеет стандартный обработчик. Его описание опущено.

Алгоритм обработки событий для третьей формы приложения.

Глобальные переменные:

hDrive,hMBRDrive: THandle; - Дескрипторы дисков для работы с главной загрузочной записью носитель и блоком параметров Bios.

DiscGeometry:TDiscGeometry; - Структрура DiscGeometry Имеет в своём составе такие поля как количество секторов на дорожке , количество дорожек на носителе , количество байт в секторе,

1. Описание события С1. Данное событие происходит при нажатии на кнопку

«Начать работу с диском». Если дескрипторы дисков уже существуют, то происходит их уничтожение. Далее происходит создание новых дескрипторов дисков. Если дескрипторы создаются успешно, то происходит заполнение структуры Disk Geometry, в полях которой содержатся данные о диске и вывод их в соответствующее поле. Если дескрипторы создать не удалось, то выводится сообщение об ошибке.

Алгоритм обработки данного события представлен на схеме1.



Схема 1 - Алгоритм работы обработчика события С1.

1. Описание события С2: Данное событие происходит по нажатию на кнопку

«Считать BPB», после этого происходит обработка этого события. Данные из дескриптора диска читаются в буфер, эти данные располагаются в первом секторе логического диска. Если данные считаны, то происходит заполнение специальных таблиц данными из этого буфера, если нет, то выводится специальное сообщение о том, что данные прочитать не удалось.

Алгоритм обработчика этого события представлен на схеме2.



Схема2 –Обработчик события С2. «Считывание блока параметров BIoS»

1. Описание события С3. Данное событие аналогично предыдущему, за исключением сектора который необходимо прочитать. Данное событие реализуется при нажатии на кнопку «считать Partition Table». Конструктивно данный алгоритм идентичен предыдущему – данные из первого сектора носителя данных читаются в буфер, если они успешно прочитаны происходит заполнение соответствующих страниц данными, если нет, то выводится сообщение о том, что данные не удалось считать в буфер. Алгоритм обработчика события представлен на схеме3.



*Дополнительные структуры и алгоритмы*

В данном разделе будут рассмотрены дополнительные логические структуры приложения:

1. Алгоритм работы потока «приёмника» (программно ThreadReceiver)
2. Алгоритм работы модуля, определяющего все активные носители, присутствующие в системе.(программно расположена в модуле HDDInfo).
3. Алгоритм работы потока «приёмника» представлен на схеме 4. Описание алгоритма: В потоке организуется бесконечный цикл(пока поток не остановлен) в этом цикле проверяется некоторое событие, сигнализирующее потоку о том, что данные записаны в файл. После того как событие перешло в сигнальное состояние происходит его обратный сброс в несигнальное состояние. Затем происходит проецирование файла, переменная которого уже создана при запуске основной программы, на адресное пространство процесса. Если проекция создана, то данные читаются из файла и передаются в основную форму программы, а точнее в поле истории переписки. Если создать проекцию не удалось то появляется соответствующее сообщение.



Схема 4 – Алгоритм работы потока «приёмника».

1. Описание алгоритма работы модуля, определяющего все активные носители в системе. Алгоритм работы представлен на схемах 5,6,7 и 8.

1. Описание алгоритма (схема5): При выполнении функции, реализующей данный алгоритм, в её теле происходит создание массива имён диско – массив латинских букв A-Z. После этого в цикле происходит считывание имён дисков присутствующих в системе и сравнивание этих имён с именами из массива. Если носитель с данным именем присутствует, то это имя возвращается в качестве результата работы функции.



Схема 5 –Алгоритм получения имени носителя, присутствующего в системе.

2. Описание алгоритма (схема6 ): Данный алгоритм необходим для верного определения номера системного носителя. Входными параметрами для этой функции является номер логического диска полученный с помощью API функции GetVolumeInformation. Данный номер считывается в строку и длина этой строки присваивается специальной переменной. Если значение этой переменной не равно восьми, происходит выход из процедуры, если и значение номера не меняется, если длина равна 8, то в цикле происходит изменение номера, считанного с помощью функции GetVolumeInformation, на правильный. В семействе ОС Windows в поздних версиях таких как «XP», «Vista» и «7» номер, полученный с помощью функции GetVolumeInformation является общим для всех носителей присутствующих в системе, хотя на практике это не является таковым.



Схема 6 –алгоритм работы функции, определяющей верный номер логического диска.

Данная функция необходима для правильного определения номера логического диска, если программа запущена под операционной системой Windows Xp , или более поздних.



Схема 7-Алгоритм процедуры для получения системной информации о логическом носителе.

Входным параметром для данной функции является имя логического диска, которое извлекается при вызове функции : GetDisk, алгоритм которой представлен на схеме 5.

3. Описание алгоритма (схема 8). Этот алгоритм реализуется при вызове функции, определяющей все присутствующие в системе диски, при запуске третьей формы программы(этап3). В данном алгоритме реализуется, сперва, вызов функции GetDisks, для получения имён всех дисков присутствующих в системе и записи этих имён в строку, далее в цикле происходит считывание символа из строки и получение с помощью функции GetVolumeInformation всех данных о диске с именем совпадающем со считанным символом. Если данные считаны , то происходи вывод их в специальную компоненту, если нет , то происходит заполнение полей отображающих данные о носителе нулевыми значениями.



Схема 8 – Алгоритм работы функции вызова процедуры получающей данные о всех активных носителях системы.

**2.3.2 Логическая структура справочной системы.**

Сначала в файле «текст справки.rtf» в редакторе MS Word составляется содержание разделов справки. Затем с помощью программы Microsoft Help Workshop создается оглавление справочной системы и проект. В оглавлении вводятся названия всех разделов и номера. В проекте указывается файл rtf, файл содержания, идентификаторам разделов сопоставляются номера и названия, устанавливаются настройки и производится компиляция. После компиляции создается файл справочной системы CПРАВКА.HLP.

Рисунок 3.1 - Пример раздела в файле «текст справки.rtf».

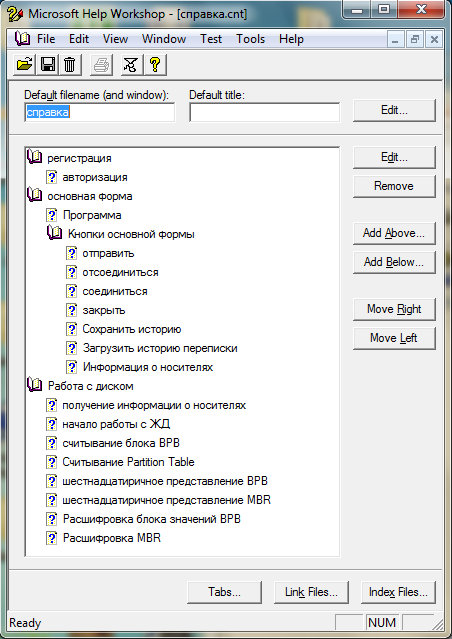
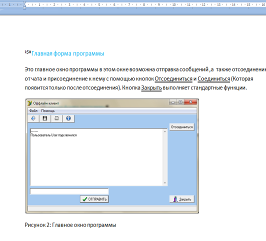


Рисунок 3.2 - Пример файла содержания справочной системы

.

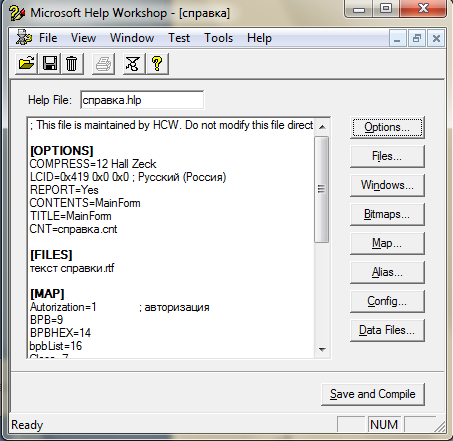


Рисунок 3.3 - Пример файла проекта справочной системы

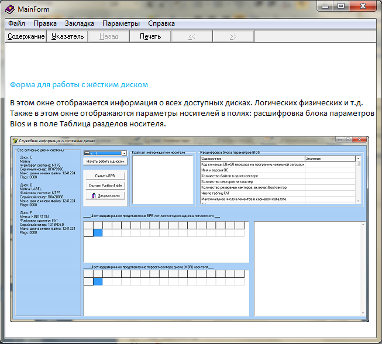


Рисунок 4.4 Пример окна справки.

**2.3.3 Создание программы инсталлятора**

Для создания инсталлятора использовалась программа InstallShield Express.

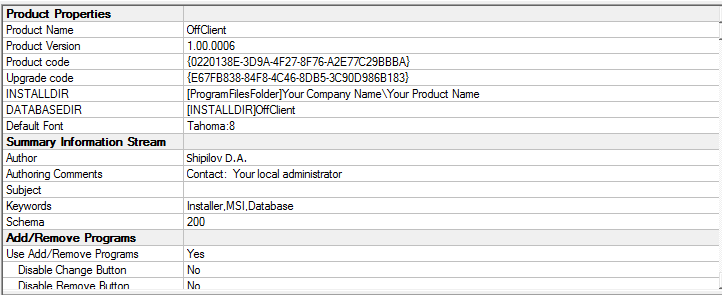


Рисунок 3.3.1 – Общая информация.

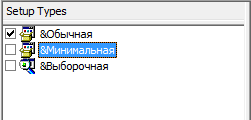


Рисунок 3.3.2 - Тип установки.

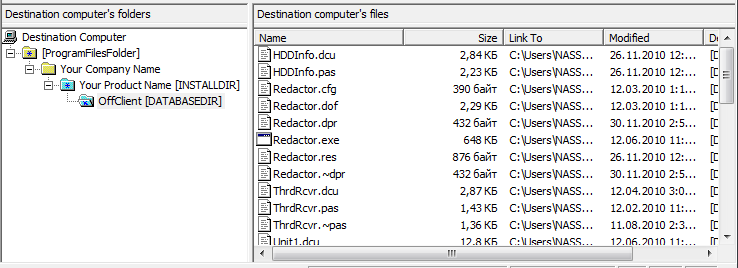


Рисунок 3.3.4 - Содержание устанавливаемых файлов проекта

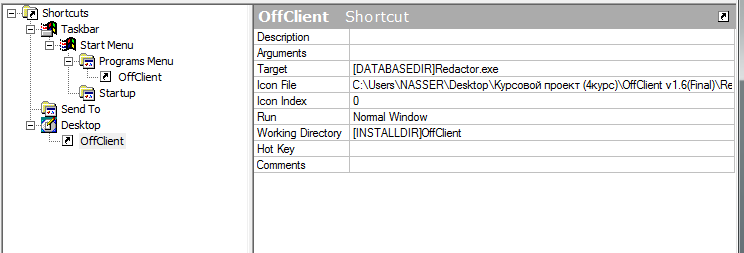


Рисунок 3.3.5 - Создание ярлыков

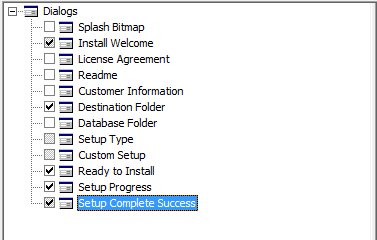


Рисунок 3.3.6 - Выбор диалоговых окон.

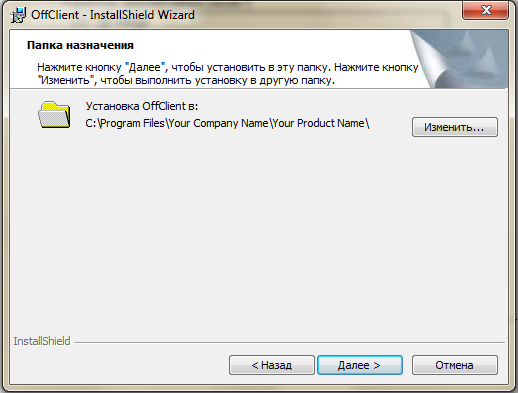


Рисунок 3.3.7 - Программа установки.

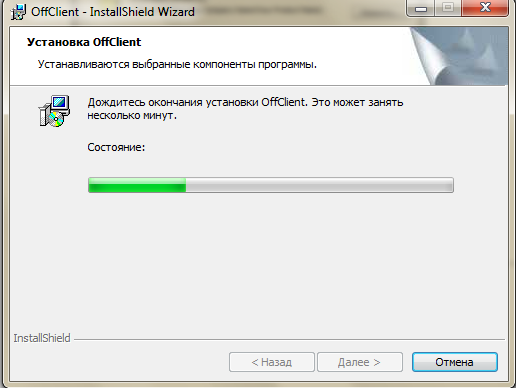


Рисунок 3.3.8 - Ход установки.

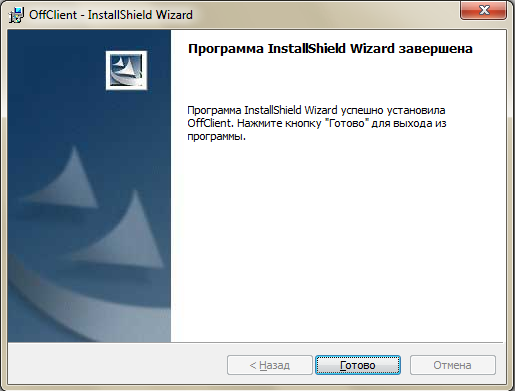


Рисунок 3.3.9 - Окончание установки.

## 2.4 Используемые технические средства

При выполнении данной курсовой работы используется персональный компьютер операционной системой Windows7 с процессором IntelCore2Duo6300. Данный компьютер оснащен всеми устройствами необходимыми для работы в среде разработки Delphi 7 и последующем запуске полученных программ.

## 2.5 Вызов и загрузка

Вызов программы осуществляется выполнением приложения OffClient, ярлыки которого находятся на рабочем столе и панели программ Windows.

## 2.6 Входные данные

Входными данными в данной программе являются сообщения ,которые вы вводите в поле ввода сообщений. Также входными данными являются имена логических дисков, которые выбираются из выпадающего списка.

## 2.7 Выходные данные

Выходными данными в данном приложении являются:

Для первого задания это сообщения пришедшие от пользователей других копий программы, а также сообщения, которые были посланы из текущей версии приложения

Для второго задания выходными данными являются данные о системном носителе, который выбирается из списка доступных носителей. Эта информация отображается в полях и таблицах второй части программы.

# 3. Программа и методика испытаний

# 

# 3.1 Объект испытаний

Объектом испытаний является приложение Redactor.exe.

## 

# 3.2 Цель испытаний

Целью испытании приложения является выяснение возможных ошибок во время выполнения программы, которые не были обнаружены на стадии разработки.

# 3.3 Требования к программе

Требуется проверить работу логической и интерфейсной частей программ. А так же обеспечить контроль входных и выходных данных.

# 3.4 Требования к программной документации

В приложении необходимо запустить справочную систему.

# 3.5 Средства и порядок испытаний

Испытание приложения будет производиться 2-мя методами: “белым и черным ящиками”.

# 3.6 Методы испытаний

# 3.6.1 Тестирование методом белого ящика:

Прежде чем нумеровать текст программы, необходимо все циклы изменить на структурированные, сложные условия разбить на простые, а линейные части программы(по возможности) записать в одну строку. Затем строки текста функции нумеруются: номер получают только те строки, которые содержат операторы, условия или закрывающие операторные скобки.

1. Тестирование процедуры выполняющей запись данных в проецируемый файл(2 этап выполнения программы). В данном случае это является первым заданием курсового проекта - обмен данными между приложениями с помощью файла проецируемого в память.

1инициализация

2if Edit1.Text <> '' then

Begin

WriteDataInMMF(Nicname+' Написал:'+#13#10+string(Edit1.Text));

(тело функции WriteDataInMMF(вхдными данными является строка S)

имеет следующий вид):

begin

3 WaitForSingleObject(HMutex, INFINITE);

PBaseAdress:=MapViewOfFile(HFileSender,FILE\_MAP\_WRITE,0,0,Length(s)+4);

4 if(PBaseAdress = nil) then

begin

5 CloseHandle(HFileSender);

st:='не удалось передать данные';

Application.MessageBox(st,'Сообщение...',MB\_OK);

exit;

end;

begin

6 integer(PBaseAdress^):=length(s);

CopyMemory(Pointer(Integer(PBaseAdress)+4),PChar(s),length(s));

UnmapViewOfFile(PBaseAdress);

SetEvent(HEvent);

ReleaseMutex(HMutex);

7 end;

8 end;

9 Edit1.Text:='';

10 end

else

11 exit;

12 end;



Рисунок 4.1 – Потоковый граф обработчика события «Событие1» которое происходит при нажатии на кнопку «ОТПРАВИТЬ» на основной форме программы.

Цикломатическая сложность потокового графа:

1) V(G) = 3 региона;

2) V(G) = 13 дуг - 12 узлов + 2 = 3;

3) V(G) = 2 предикатных узла +1 = 3.

Базовое множество независимых линейных путей:

**Путь 1**:1-2-11-12 .

**Путь 2**: 1-2-3-4-5-12.

**Путь 3**: 1-2-3-4-5-6-7-8-9-10-12.

Таблица тестовых вариантов 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| №  пути | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Строка содержащая, которая должна содержать текст сообщения пуста | Не происходит отправки пустого сообщения | Состояние программы после перехода на главную форму не |
| 2 | Не удалось отобразить файл на адресное пространство процесса при обращении к дескриптору файла , проецируемого в память. | Сообщение:  «Не удалось отобразить файл на адресное пространство процесса».. | Сообщение:  «Не удалось отобразить файл на адресное пространство процесса». |
| 3 | Текст сообщения: «Привет!» | Появление текста «Привет!» в поле истории переписки. | Удачный исход – появление текста «Привет!» в поле истории переписки |

1. Тестирование процедуры выполняющей создание дескрипторов логического диска с

которым ведётся работа и дескриптора жёсткого диска которому принадлежит этот логический диск(3 этап выполнения программы ).

1 инициализация

2 if hDrive <> 0 then

3CloseHandle(hDrive);

4 if hMBRDrive <> 0 then

5CloseHandle(hMBRDrive);

6 discNameBPB:='\\.\'+DriveComboBox1.Drive+':';

discNameMBR:='\\.\PHYSICALDRIVE'+intToStr(0);

hMBRDrive:= CreateFile(PChar(discNameMBR),GENERIC\_READ, FILE\_SHARE\_READ or FILE\_SHARE\_WRITE,nil,OPEN\_EXISTING,0,0);

hDrive:= CreateFile(PChar(discNameBPB),GENERIC\_READ, FILE\_SHARE\_READ or FILE\_SHARE\_WRITE,nil,OPEN\_EXISTING,0,0);

7 if (hDrive <> INVALID\_HANDLE\_VALUE) and (hMBRDrive <> INVALID\_HANDLE\_VALUE) then

begin

8 Result := DeviceIoControl(hDrive,IOCTL\_DISK\_GET\_DRIVE\_GEOMETRY,nil,0,

@DiscGeometry,SizeOf(TDiscGeometry),junk,nil) and (junk = SizeOf(TDiscGeometry));

DiscGeometryShow();

9 end

else

begin

10 ShowMessage('не удалось создать дескрипторы дисков');

11 exit;

end;

12 end;



Рисунок 4.2 Потоковый граф обработчика события «С1» которое происходит при нажатии на кнопку «Начать работу с диском»

Цикломатическая сложность потокового графа:

1) V(G) = 4 регионов;

2) V(G) = 14 дуга - 12 узлов + 2 = 4;

3) V(G) = 3 предикатных узлов +1 = 4.

**Путь 1**:1-2-4-6-7-10-11-12;

**Путь 2**:1-2-3-4-6-7-10-11-12.

**Путь 3**: 1-2-4-5-6-7-10-11-12.

**Путь 4**:1-2-3-4-5-7-8-9-12.

|  |  |  |  |
| --- | --- | --- | --- |
| №  пути | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Дескрипторы файлов не созданы . | Невозможно получить данные о носителе вывод сообщения: «Не удалось создать дескрипторы файлов» | Вывод сообщения  «Не удалось создать дескрипторы файлов» |
| 2 | Создан дескриптор файла BPB | Невозможно получить данные о носителе вывод сообщения: «Не удалось создать дескрипторы файлов» | Вывод сообщения  «Не удалось создать дескрипторы файлов» |
| 3 | Создан дескриптор файла MBR | Невозможно получить данные о носителе вывод сообщения: «Не удалось создать дескрипторы файлов» | Вывод сообщения  «Не удалось создать дескрипторы файлов» |
| 4 | Дескрипторы файлов существуют как MBR так и BPB | Загрузка данных о носителе. | Загрузка данных о носителе. |

1. Тестирование процедуры считывающей блок параметров Bios - событие «С2» (происходит при нажатии на кнопку «Считать BPB»)логического диска

дескриптор которого был создан в процедуре обрабатывающей событие «С1» при нажатии на кнопку «Начать работу с диском»

1инициализация

2IfReadSectors(hDrive,0,1,@buffer,DiscGeometry.BytesPerSector) = DiscGeometry.BytesPerSector then

begin

3for i:= 1 to 16 do

4Grid1.Cells[i,0]:=intToHex((i-1),1);

5i:=1; j:=1; k:=1;

6while k<=DiscGeometry.BytesPerSector do

begin

7 if i=1 then

8 Grid1.Cells[0,j]:=IntToHex(((j-1)\*16+(i-1)),3);

9 Grid1.Cells[i,j]:=IntToHex(Buffer[k],2);

inc(k);

inc(i);

10 if i>16 then

begin

11 i:=1;

j:=j+1;

Grid1.RowCount:= Grid1.RowCount+1;

12end;

13end;

14{линейный блок расшифровки данных BPB}

end

else

15ShowMessage(‘не удалось считать BPB’);

16end

17end;

программа многопоточный приложение служебный



Рисунок 4.3 – Потоковый граф обработчика события С2 (Считывание блока параметров Bios).

Цикломатическая сложность потокового графа:

1) V(G) = 6 регионов;

2) V(G) = 21дуг - 17 узлов + 2 = 6;

3) V(G) = 5 предикатных узлов +1 = 6.

**Путь 1**:1-2-15-16-17;

**Путь 2**:1-2-3-4-5-6-7-9-10-12-13-14-17.

**Путь 3**: 1-2-3-4-5-6-7-8-9-10-12-13-14-16-17.

**Путь 4**:1-2-3-4-5-6-7-9-10-11-12-13-14-16-17.

**Путь 5**:1-2-3-4-5-6-7-8-9-10-11-12-13-14-16-17.

**Путь 6**: 1-2-3-4-3-4-5-6-7-9-10-12-13-6-7-8-9-10-11-12-13-14-16-17.

|  |  |  |  |
| --- | --- | --- | --- |
| №  пути | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Не удалось считать сектор BPB . | Вывод сообщения о том, что считать BPB не удалось | Вывод сообщения: « не удалось считать BPB» |
| 2 | Часть данных считана , при этом номер строки остаётся прежним и номер элемента строки не достиг 16. | Вывод данных в таблицу, содержащую данные о BPB,например команду перехода на программу загрузки: «EB0090h» | Вывод данных в таблицу, содержащую данные о BPB: команда перехода на программу загрузки: «EB0090h» |
| 3 | Чтение данных началось и строка получила значение | Вывод данных в таблицу, содержащую данные о BPB,например команду перехода на программу загрузки: «EB0090h» | Вывод данных в таблицу, содержащую данные о BPB: команда перехода на программу загрузки: «EB0090h» |
| 4 | Строка считана и номер элемента в строке достиг 16 | Вывод данных в таблицу, содержащую данные о BPB,например команду перехода на программу загрузки: «EB0090h» | Вывод данных в таблицу, содержащую данные о BPB: команда перехода на программу загрузки: «EB0090h» |
| 5 | Строка получила номер и элемент в строке достиг 16 | Вывод данных в таблицу, содержащую данные о BPB,например команду перехода на программу загрузки: «EB0090h» | Вывод данных в таблицу, содержащую данные о BPB: команда перехода на программу загрузки: «EB0090h» |
| 6 | Чтение данных из буфера в течении нескольких проходов циклов с изменением номера строк и столбцов таблицы , содержащей значения BPB | Пошаговый вывод данных в таблицу содержащую 16рмчные значения и вывод данных в таблицу содержащую расшифрованные значения | Обе таблицы «Шестнадцатиричное представление BPB логического диска» и «Расшифровка блока параметров BIOS» заполняются необходимыми значениями |

1. Тестирование процедуры, считывающей таблицу разделов носителя(событие «С3» - происходит при нажатии на кнопку «Считать Partition Table»), к которому принадлежит ,выбранный из выпадающего списка логический диск

1инициализация.

2Memo2.Clear;

3If.ReadSectors(hMBRDrive,0,1,@buffer,DiscGeometry.BytesPerSector)= DiscGeometry.BytesPerSector then

begin

4for i:= 1 to 16 do

5Grid2.Cells[i,0]:=intToHex((i-1),1);

6i:=1; j:=1; k:=1;

7while k<=DiscGeometry.BytesPerSector do

begin

8if i=1 then

9Grid2.Cells[0,j]:=IntToHex(((j-1)\*16+(i-1)),3);

10Grid2.Cells[i,j]:=IntToHex(Buffer[k],2);

inc(k);

inc(i);

11if i>16 then

begin

12 i:=1;

j:=j+1;

Grid2.RowCount:= Grid2.RowCount+1;

13 end;

14end;

begin

15{Линейный блок расшифровки значений полученных в буфере после чтения сектора}

16end

else

17ShowMessage('Не удалось считать блок PartitionTable');

18end;



Рисунок 4.4 – Потоковый граф обработчика события «С3» (кнопка «Считать PartitionTable »)

Цикломатическая сложность потокового графа:

1) V(G) = 6 регионов;

2) V(G) = 22 дуга - 18 узлов + 2 = 6;

3) V(G) = 5 предикатных узлов +1 = 6.

**Путь 1**:1-2-17-18;

**Путь 2**:1-2-3-4-5-6-7-8-10-11-13-14-15-16-18;

**Путь 3**:1-2-3-4-5-6-7-8-9-10-11-13-14-15-16-18;

**Путь 4**: 1-2-3-4-5-6-7-8-10-11-12-13-14-15-16-18;

**Путь 5**: 1-2-3-4-5-4-5-6-7-8-9-10-11-13-14-15-16-18;

**Путь 6**: 1-2-3-4-5-4-5-6-7-8-10-11-12-13-14-7-8-9-10-11-13-14-15-16-18;

|  |  |  |  |
| --- | --- | --- | --- |
| № пути | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Если не удалось считать Partition Table | Должно появиться сообщение о том , что не возможно считать Partition Table | Вывод сообщения:  «Не удалось считать Partition Table» |
| 2 | Происходить считывание элемента массива , который имеет номер отличающийся от 1 и от 16. | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» |
| 3 | Считывание элемента массива , который имеет номер 1 в строке таблицы | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» |
| 4 | Считывание элемента, который имеет последний номер в строке (16) | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» |
| 5 | Считывание данных из массива. При прохождении этого пути проверяется заполнение верхней строчки таблицы номерами от 0 до F. | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» |
| 6 | Считывание данных из массива. Когда элемент имеет номер 16 в строке, а последующий элемент имеет номер 1 в след. строке. | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» | Вывод данных, например: «Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }» |

*Дополнительный тест*

Тестирование алгоритма работы потока, который также относится к основному заданию курсового проекта.

1инициализация.

2while (not terminated) do

begin

3 WaitForSingleObject(Form1.HEvent, INFINITE);

4 ResetEvent(Form1.HEvent);

5 try

// резерв. память в потоке для этотой проекции

6PBaseAdress:=MapViewOfFile(Form1.HFileSender, FILE\_MAP\_READ,0,0,65536);

7 if (PBaseAdress = nil) then

begin

8 CloseHandle(Form1.HFileSender);

st:='не удалось отобразить файл на адресное пространство';

Application.MessageBox(st,'Сообщение...',MB\_OK);

form1.BitBtn2.Visible:=false;

form1.Button1.Visible:=false;

9 exit;

10end;

11 nlineSize:=integer(PBaseAdress^);

SetLength(TextString,nlineSize);

CopyMemory(PChar(TextString),Pointer(Integer(PBaseAdress)+4),nlineSize);

Form1.Memo1.Text:=Form1.Memo1.Text+'---------'+#13#10;

Form1.Memo1.Text:=Form1.Memo1.Text+TextString+#13#10;

sendmessage(Form1.memo1.handle, em\_linescroll, 0, Form1.memo1.lines.count-1);

UnmapViewOfFile(PBaseAdress);

finally

12end;

13end;



Рисунок 4.5 – Потоковый граф обработчика потока(данный обработчик выполняется постоянно после запуска программы и если не была нажата кнопка «отсоединится при работе программы»)

Цикломатическая сложность потокового графа:

1) V(G) = 3 регионов;

2) V(G) = 14 дуг - 13 узлов + 2 = 3;

3) V(G) = 2 предикатных узлов +1 = 3.

**Путь 1**:1-2-3-4-5-6-7-8-9-13;

**Путь 2**:1-2-3-4-5-6-7-10-11-12-13;

**Путь 3**:1-2-3-4-5-6-7-8-9-10-11-12-2-3-4-5-6-7-10-11-12-13;

|  |  |  |  |
| --- | --- | --- | --- |
| №  пути | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Не удалось отобразить файл на адресное пространство в потоке. | Должно появиться сообщение : не удалось отобразить данные на адресное пространство потока | Вывод сообщения:  «Не отобразить файл на адресное пространство» |
| 2 | Ввод данных в поле сообщения и однократное нажатие на кнопку «отправить» | Вывод сообщения «отправить» в поле истории переписки. | Сообщение введённое в поле сообщений «отправить» появилось в поле истории переписки. |
| 3 | Ввод сообщения в поле сообщений и нажатие на кнопку «отправить» с последующим повторение этих действий | Вывод сообщения «отправить» в поле истории переписки. | Сообщение введённое в поле сообщений «отправить» появилось в поле истории переписки. |

**3.6.2 Тестирование методом черного ящика**

1. Тестирование процедуры, выполняющей запись данных в проецируемый файл

(Рисунок 4.6. ).

(Этап выполнения программы 2 «Событие 1»).

Предусловие : Текст сообщения должен быть обязательно введён.

Постусловие: Если проекция файла создана, и объект типа поток сработал нормально, то вывод текста сообщения в поле истории переписки.



Рисунок 4.6 – Дерево разбиений для события «Событие1» (первое задание)

|  |  |  |  |
| --- | --- | --- | --- |
| № листа | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Сообщение не введено | Никаких действий не должно происходить, программа не должна отвечать на действия такого рода | Программа никак не реагирует |
| 2 | Проекция файла не создана | Вывод сообщения: «Не удалось передать данные» | Вывод сообщения: «Не удалось передать данные» |
| 3 | Поток не сработал | Вывод сообщения : «Не удалось создать проекцию файла» | Вывод сообщения : «Не удалось создать проекцию файла» |
| 4 | В поле ввода сообщения введено : «СЛОВО» | Вывод сообщения с содержимым «СЛОВО» в поле истории переписки | В поле истории появится сообщение с текстом :  «<имя введённое при авторизации> написал :  СЛОВО» |

1. Тестирование процедуры выполняющей создание дескрипторов носителя с которым ведётся работа (Рисунок 4.7).

(Этап выполнения программы 3 , событие С1).

Предусловие: Программа должна быть запущена от имени администратора.

Постусловие: 1 Должен быть выбран присутствующий носитель.

2 Если дескрипторы файлов успешно созданы и данные прочитаны в буфер, то- вывод информации в таблицы.



Рисунок 4.7 Дерево разбиений для процедуры, выполняющей создание дескрипторов носителя(Этап выполнения программы 3 события C1).

|  |  |  |  |
| --- | --- | --- | --- |
| №  листа | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Программа запущена не от имени администратора | Заполнение таблиц ,содержащих данные о BPB и данные о Partition Table нулевыми значениями. | Заполнение таблиц нулевыми значениями |
| 2 | Выбран носитель отсутствующий в списке активных носителей | Заполнение таблиц ,содержащих данные о BPB и данные о Partition Table нулевыми значениями. | Заполнение таблиц нулевыми значениями |
| 3 | Если не удалось создать дескрипторы файлов, отождествляющих жёсткий и логический диски. | Вывод сообщения :  «Не удалось создать дескрипторы носителя» | Вывод сообщения :  «Не удалось создать дескрипторы носителя» |
| 4 | Выбран активный носитель и дескрипторы носителя успешно созданы. | Вывод данных о блоке параметров Bios , Partition Table и расшифрованных значений в специальные таблицы. | Заполнение таблиц всеми необходимыми данными о носителе. |

3)Тестирование процедуры, выполняющей считывание блока параметров Bios (Рисунок 4.8).

(Этап выполнения программы 3 , событие «С2»).

Предусловие: Дескрипторы должны быть созданы.

Постусловие: Файловая система должна быть FAT32.



Рисунок 4.8 –Дерево разбиений для процедуры, выполняющей считывание блока BPB для выбранного носителя.

|  |  |  |  |
| --- | --- | --- | --- |
| №  листа | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Дескрипторы не созданы. | Вывод сообщения «Не удалось считать блок параметров Bios» | Вывод сообщения «Не удалось считать блок параметров Bios» |
| 2 | Файловая система отличается от FAT32. | Заполнение таблицы расшифровки BPB неверными значениям , например : «Количество FAT : 0» | Заполнение таблицы содержащей расшифрованные данные о BPB неверными значениям: «Количество FAT : 0» |
| 3 | Файловая система FAT32 | Заполнение таблицы расшифровки верными значениями, например: «количество FAT : 2» | Заполнение таблицы расшифровки верными значениями: «количество FAT : 2» |

4)Тестирование процедуры, считывающей таблицу разделов жёсткого диска(Рисунок 4.9).

(Этап выполнения программы 3 , событие «С3»).

Предусловие: Программа должна быть запущена от имени администратора.

Постусловие: Дескрипторы должны быть созданы.



Рисунок 4.9 – Дерево разбиений для процедуры считывающей таблицу разделов носителя.

|  |  |  |  |
| --- | --- | --- | --- |
| №  листа | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Программа запущена не от имени администратора. | Заполнение полей таблицы расшифровки нулевыми значениями. | Заполнение полей таблицы расшифровки нулевыми значениями. |
| 2 | Дескрипторы не созданы | Вывод сообщения:  «Не удалось считать PartitionTable» | Вывод сообщения:  «Не удалось считать PartitionTable» |
| 3 | Дескрипторы созданы | Заполнение таблицы расшифровки верными значениями, например: «Размер раздела {: 61432497 }» | Заполнение таблицы расшифровки верными значениями «Размер раздела {: 61432497 }» |

5)Тестирование процедуры работы потока.(Рисунок 4.10).

Предусловие: 1.Поток должен быть запущен.

2.Средства синхронизации созданы.

Постусловие: Файл проецируемый в память должен быть успешно создан.



Рисунок 4.10 – Дерево разбиений для процедуры работы потока.

|  |  |  |  |
| --- | --- | --- | --- |
| №  листа | Исходные данные | Ожидаемые результаты | Реальные результаты |
| 1 | Средства синхронизации не созданы. | Вывод сообщения «не удалось осуществить обмен сообщениями» | Вывод сообщения «не удалось осуществить обмен сообщениями» |
| 2 | Проекция файла не создана. | Вывод сообщения «не удалось отобразить файл на адресное пространство» | Вывод сообщения «не удалось отобразить файл на адресное пространство» |
| 3 | Проекция файла создана.  Было введено сообщение  «Отправлено» | Вывод сообщения :  «Отправлено» | Вывод сообщения :  «Отправлено» |

**4. Эксплуатационные документы**

**4.1 Руководство программиста**

**4.1.1 Общие сведения о программе**

Программа предназначена для обмена сообщениями между копиями этой программы с помощью файла, проецируемого в память. А также для анализа рабочей области системного диска, т.е анализ и расшифровку блока параметров Bios и анализ и расшифровку таблицы разделов носителя.

**4.1.2 Структура программы**

После запуска приложения появляется форма «Авторизация» для присвоения пользователя специального псевдонима для обмена информацией. Форма содержит две кнопки: «Войти» и «Закрыть». По нажатию кнопки «закрыть» происходит выход из приложения. По нажатию кнопки «войти» происходит переход на основную форму программы, данная форма содержит поле вывода сообщений – «поле истории переписки», также поле ввода сообщений, кнопку «ОТПРАВИТЬ» для отправки сообщений, а также кнопки отсоединиться, закрыть и соединится, которая будет появится на форме только после нажатия на кнопку «отсоединиться». Также на форме присутствует панель инструментов с кнопками :

1. Сохранить историю переписки,
2. Загрузить историю переписки.
3. Получить данные о носителях.
4. Справка.

При нажатии на первую кнопку появляется диалоговое окно, в котором необходимо выбрать файл, содержащий историю переписки, если такой имеется, и вывод содержимого этого файла в поле истории переписки.

При нажатии на вторую кнопку происходит открытие диалогового окна, в котором предлагается назначить сохраняемому файлу имя и каталог, в котором данный файл будет сохранён. Содержимым данного файла будет являться история последней переписки.

При нажатию на четвёртую кнопку происходит запуск файла справочной системы, содержащей всю необходимую информацию о программе.

При нажатии на третью кнопку происходит открытие дополнительного окна программы.

Дополнительное окно программы имеет заголовок «Служебная информация о носителях». Это окно содержит поле вывод информации о всех носителях присутствующих в системе . Поле вывода краткой информации о носителе, поле вывода расшифрованной информации о блоке параметров Bios, поле вывода расшифрованной информации о таблице разделов носителя, а также две таблицы, содержащие 16ричные значения первого сектора логического диска- блок параметров Bios и 16ричные значения первого сектора носителя, к которому относится выбранный логический диск (в этом секторе находится таблица разделов логического диска – Partition Table). Также на форме находятся 4 кнопки:

1. Начать работу с диском
2. Считать BPB(блок параметров Bios)
3. Считать Partition Table.
4. Закрыть.

При нажатии на первую кнопку происходит вывод краткой информации о носителе. При нажатии на кнопку «Считать BPB» происходит считывание блока BPB и заполнение значениями таблицы содержащей расшифрованные данные из BPB и таблицы, содержащей 16ричные значения считанные из BPB, выбранного логического диска.

При нажатии на кнопку – «Считать Partition Table» происходит считывание первого сектора носителя информации и заполнение данными таблицы, содержащей расшифрованные данные о таблице разделов диска и таблицы содержащей 16ричные значения считанные из первого сектора носителя.

**4.1.3 Проверка программы**

**Входные данные :** строка сообщения «сообщение №1»; логический диск «H».

**Выходные данные** : «сообщение№1»;

Краткая информация о носителе:

«Вы просматриваете логический диск: h

Количество цилиндров: 247

Тип носителя: съёмный носитель

Дорожек на цилиндре: 255

Секторов на дорожке: 63

Байт в секторе: 512»

Информация о таблице разделов диска:

Первый элемент таблицы разделов системного диска:

Признак загрузки: 80

Начало раздела диска{Головка: 1 Сектор: 1 Цилиндр: 0 }

Тип раздела {Ос: 7 }

Конец раздела диска{Головка: 254 Сектор: 255 Цилиндр: 255 }

Номер Сектора {: 63 }

Размер раздела {: 61432497 }

Второй элемент таблицы разделов системного диска:

Признак загрузки: 00

Начало раздела диска{Головка: 255 Сектор: 255 Цилиндр: 15 }

Тип раздела {Ос: 254 }

Конец раздела диска{Головка: 255 Сектор: 255 Цилиндр: 240 }

Номер Сектора {: 61432560 }

Размер раздела {: 318782480 }

**4.1.4 Дополнительные возможности**

Программа имеет справочную систему, для её запуска необходимо нажать кнопку , или F1 на клавиатуре. Также программа определяет носители доступные в системе, это происходит автоматически при переходе на третью форму программы.



**4.1.5 Сообщения системному программисту**

1) При подключении, если не удалось создать проекцию файла при запуске программы, появляется сообщение:

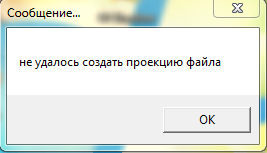


Рисунок 5.1 –Содержание сообщения

2)Если не удалось создать объект типа Mutex для обмена сообщениями с другими программами, появляется сообщение:

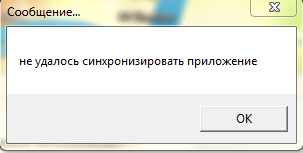


Рисунок 5.2 – Содержание сообщения.

1. Если не удалось наладить приём сообщений от других копий программ, появляется сообщение:

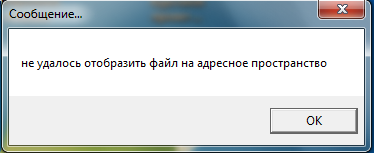


Рисунок 5.3 – Текст сообщения.

1. Если не удалось создать объект типа события для оповещения потока о том, что данные записаны в проецируемый файл, появляется сообщение:

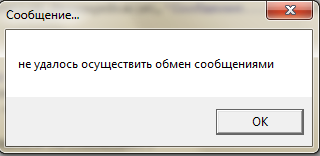


Рисунок 5.4 –Текст сообщения.

1. Если не удалось передать данные (записать данные в проецируемый файл) , появляется сообщение:

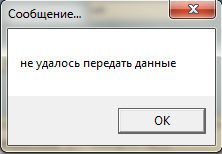


Рисунок 5.6 – Текст сообщения.

**4.2 Руководство пользователя**

**4.2.1 Назначение и условия выполнения программы**

Данная программа предназначена для обмена сообщениями между копиями данной программы, а также для получения системной информации о носителях. Для выполнения программы необходим IBM-совместимый персональный компьютер, оснащенный процессором Intel или аналогичным и операционной системой семейства Windows.

**4.2.2 Обращение к программе для запуска:**

Для запуска программы необходимо запустить приложение с Redactor.exe . Это делается с помощью ярлыка , который появляется на рабочем столе после установки, с именем OffClient.

**4.2.3 Входные и выходные данные:**

Входными и выходными данными при обмене сообщениями являются сами сообщения. При получении служебных данных о носителях входными данными являются имена этих носителей, а выходными – данные о них, такие как:

Количество цилиндров, тип носителя, дорожек на цилиндре, секторов на дорожке, байт в секторе и т.д.

**4.2.4 Сообщения оператору:**

1) Если не введён псевдоним при регистрации то появится сообщение:

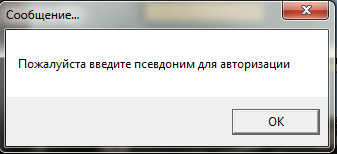


Рисунок 5.7 - Текст сообщения.

2) Если не удалось передать сообщение:

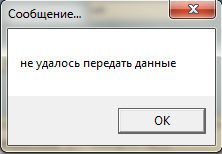


Рисунок 5.8 – Текст сообщения.

# Приложение

**1) Текст основного модуля программы**

program Redactor;

uses

Forms,

Unit1 in 'Unit1.pas' {Form1},

ThrdRcvr in 'ThrdRcvr.pas',

Unit2 in 'Unit2.pas' {Form2},

Unit3 in 'Unit3.pas' {Form3},

HDDInfo in 'HDDInfo.pas';

{$R \*.res}

begin

Application.Initialize;

Application.HelpFile:='СПРАВКА.HLP';

Application.CreateForm(TForm1, Form1);

Application.ShowMainForm:=false;

Application.CreateForm(TForm2, Form2);

Application.Run;

end.

**2) Текст модуля Unit1:**

unit Unit1; //Основной модуль

interface

uses

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

Dialogs, ThrdRcvr,

StdCtrls, Buttons, Menus,ToolWin, ComCtrls, ActnMan, ActnColorMaps,

ImgList, ExtCtrls,Unit3;

const

MMFName:PChar = 'MMFProectFile';

MutxName:PChar = 'UniqMutexName';

EvntName:Pchar = 'ChatEventName';

SHARED\_MEMORY\_SIZE = 65536;

type

TForm1 = class(TForm)

Memo1: TMemo;

OpenDialog1: TOpenDialog;

SaveDialog1: TSaveDialog;

MainMenu1: TMainMenu;

N1: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N6: TMenuItem;

N7: TMenuItem;

BitBtn2: TBitBtn;

BitBtn3: TBitBtn;

Edit1: TEdit;

Button1: TButton;

Button2: TButton;

ToolBar1: TToolBar;

OpenToolButton: TToolButton;

SaveAsToolButton: TToolButton;

ToolButton3: TToolButton;

ImageList1: TImageList;

StatusBar1: TStatusBar;

Timer1: TTimer;

ToolButton1: TToolButton;

N2: TMenuItem;

N8: TMenuItem;

procedure Button1Click(Sender: TObject);

procedure N4Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure BitBtn3Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure CloseHandles;

procedure CreateHandles;

procedure Button2Click(Sender: TObject);

procedure WriteDataInMMF(s:string);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

procedure N5Click(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

procedure ToolButton3Click(Sender: TObject);

procedure ToolButton1Click(Sender: TObject);

procedure N8Click(Sender: TObject);

procedure N2Click(Sender: TObject);

private

procedure AppHint(Sender: TObject);

public

HMutex:THandle;

ThreadRec:TThreadReceiver;

HEvent:THandle;

HFileSender:Thandle;

NicName:string;

end;

var

Form1: TForm1;

TxtFile:TextFile;// пременная файла

St:PansiChar;

lincInt:Boolean;

TxtContaningStr:String;

userName:string;

implementation

{$R \*.dfm}

//процедура для показа кртких справок

procedure TForm1.AppHint(Sender: TObject);

begin

StatusBar1.SimpleText:=Application.Hint; //для отображения ииформации в панели соостояния

end;

//закрытие всех хендлов в этой копии программы

procedure TForm1.CloseHandles();

begin

CloseHandle(HFileSender);

CloseHandle(HMutex);

CloseHandle(HEvent);

end;

// создание всех необходимых объектов

procedure TForm1.CreateHandles;

begin

// проекция файла

HFileSender:=CreateFileMapping(INVALID\_HANDLE\_VALUE, nil, PAGE\_READWRITE, 0, SHARED\_MEMORY\_SIZE,MMFName);

lincInt:=(GetLastError() <> ERROR\_ALREADY\_EXISTS);

if HFileSender = 0 then

begin

st:='не удалось создать проекцию файла';

Application.MessageBox(st,'Сообщение...',MB\_OK);

exit;

end;

// мьютекс

HMutex:=CreateMutex(nil,false,MutxName);

if HMutex = 0 then

begin

CloseHandle(HFileSender);

st:='не удалось синхронизировать приложение';

Application.MessageBox(st,'Сообщение...',MB\_OK);

exit;

end;

// событие

HEvent:=CreateEvent(nil,true,false,EvntName);

if HEvent = 0 then

begin

CloseHandle(HFileSender);

CloseHandle(HMutex);

st:='не удалось осуществить обмен сообщениями';

Application.MessageBox(st,'Сообщение...',MB\_OK);

exit;

end;

end;

// отсоединиться

procedure TForm1.Button1Click(Sender: TObject);

begin

ThreadRec.Terminate;

WriteDataInMMF('Пользователь '+NicName+' отключился');

ThreadRec.WaitFor;

ThreadRec.Free;// правильно уничтожаем поток

CloseHandles();

button2.Visible:=true;

button1.Visible:=false;

bitbtn2.Visible:=false;

end;

// загрузить историю переписки

procedure TForm1.N4Click(Sender: TObject);

begin

if OpenDialog1.Execute then

begin

AssignFile(TxtFile,OpenDialog1.FileName);

Reset(TxtFile);

Memo1.Text:=Memo1.Text +'{-----------Загруженная переписка ниже-------------}' + #13#10;

while not eof(TxtFile) do

begin

readln(TxtFile,TxtContaningStr);

Memo1.Text:=Memo1.Text + TxtContaningStr + #13#10;

end;

Memo1.Text:=Memo1.Text +'{-----------Загруженная переписка выше -------------}' + #13#10;

CloseFile(TxtFile);

end;

end;

//процедура записи данных в MMF

procedure TForm1.WriteDataInMMF(s:string);

var

PBaseAdress:Pointer;

begin

WaitForSingleObject(HMutex, INFINITE);

PBaseAdress:=MapViewOfFile(HFileSender,FILE\_MAP\_WRITE,0,0,Length(s)+4);

if(PBaseAdress = nil) then

begin

CloseHandle(HFileSender);

st:='не удалось передать данные';

Application.MessageBox(st,'Сообщение...',MB\_OK);

exit;

end;

begin

integer(PBaseAdress^):=length(s);

CopyMemory(Pointer(Integer(PBaseAdress)+4),PChar(s),length(s));

UnmapViewOfFile(PBaseAdress);

SetEvent(HEvent);

ReleaseMutex(HMutex);

end;

end;

// процедура записи данных в PBaseAdress

procedure TForm1.BitBtn2Click(Sender: TObject);

begin

if Edit1.Text <> '' then

begin

WriteDataInMMF(Nicname+' написал:'+#13#10+string(Edit1.Text));

Edit1.Text:='';

end

else

exit;

end;

// Закрытия формы

procedure TForm1.BitBtn3Click(Sender: TObject);

begin

close;

end;

//создаём хендлы и потоки

procedure TForm1.FormCreate(Sender: TObject);

begin

Memo1.Clear;

// создание потока принимающего данные из файла

ThreadRec:=TThreadReceiver.Create(false);

ThreadRec.Priority:=tpLowest;

CreateHandles();// создаем хендлы

end;

// соединение

procedure TForm1.Button2Click(Sender: TObject);

begin

CreateHandles();

ThreadRec:=TThreadReceiver.Create(false);

button2.Visible:=false;

button1.Visible:=true;

bitbtn2.Visible:=true;

WriteDataInMMF(NicName+' Cоединился!');

end;

// Запрос подтверждения при закрытии формы

procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

case messageBox(Handle,'Действительно выйти?','Внимание!', MB\_YESNO) of

IDYES:

begin

if Button1.Visible = true then

begin

ThreadRec.Terminate;

WriteDataInMMF('Пользователь '+NicName+' отключился');

ThreadRec.WaitFor;

ThreadRec.Free;

CanClose:=true;

end

else

CanClose:=true;

end;

IDNO: CanClose:=false;

end;

end;

//Сохранить историю переписки

procedure TForm1.N5Click(Sender: TObject);

begin

If SaveDialog1.Execute then

begin

AssignFile(TxtFile,SaveDialog1.FileName);

rewrite(TxtFile);

TxtContaningStr:= Memo1.Text;

Write(TxtFile,TxtContaningStr);

CloseFile(TxtFile);

St:='Файл сохранён';

Application.MessageBox(St,'Сообщение...',MB\_OK);

exit;

end

else

begin

St:='файл не сохранён';

Application.MessageBox(St,'Сообщение...',MB\_OK);

end;

end;

procedure TForm1.Timer1Timer(Sender: TObject);

begin

if Form1.Visible = true then

begin

Application.OnHint:=AppHint;

Timer1.Enabled:=false;

Timer1.Destroy;

end;

end;

procedure TForm1.ToolButton3Click(Sender: TObject);

begin

Application.CreateForm(TForm3, Form3);

Application.ShowMainForm:=false;

Form3.Visible:=true;

end;

procedure TForm1.ToolButton1Click(Sender: TObject);

begin

//вызов справки (основная форма программы)

Application.HelpContext(2);

end;

procedure TForm1.N8Click(Sender: TObject);

begin

Application.HelpCommand(HELP\_FINDER,0);

end;

procedure TForm1.N2Click(Sender: TObject);

begin

ShowMessage('Программу подготовил студент группы 742:'+#13#10+'Шипилов Д.А.');

end;

end.

**3) Текст модуля Unit2 (Модуль формы «Авторизация»):**

unit Unit2;

interface

uses

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

Dialogs, StdCtrls,Unit1, ComCtrls, Buttons;

type

TForm2 = class(TForm)

Button2: TButton;

Edit1: TEdit;

Label1: TLabel;

BitBtn1: TBitBtn;

procedure Button2Click(Sender: TObject);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

procedure BitBtn1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

formClose:boolean=false;

implementation

{$R \*.dfm}

// закрытие формы

procedure TForm2.Button2Click(Sender: TObject);

begin

close;

end;

// запрос подтверждения при завершении программы из дочерней формы

procedure TForm2.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

if formClose = true then

begin

CanClose:=true;

end

else

begin

case messageBox(Handle,'Действительно выйти?','Внимание!', MB\_YESNO) of

IDYES:

begin

Application.Terminate;

end;

IDNO: CanClose:=false;

end;

end;

end;

procedure TForm2.BitBtn1Click(Sender: TObject);

var

St:PansiChar;

begin

if Edit1.Text <> '' then

begin

formClose:=true;

Form1.Memo1.Clear;

Form1.NicName:=Edit1.Text;

Form1.WriteDataInMMF('Пользователь '+Form1.NicName+' подключился');

Application.ShowMainForm:=true;

Form1.Visible:=true;

Form2.Close;

end

else

begin

st:='Пожалуйста введите псевдоним для авторизации';

Application.MessageBox(st,'Сообщение...',MB\_OK);

exit;

end;

end;

end.

**4) Модуль Unit 3 (Модуль формы, осуществляющей работу с носителями информации в системе):**

unit Unit3;

interface

uses

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

Dialogs, StdCtrls, FileCtrl, ExtCtrls, Grids, ValEdit,HDDInfo, Buttons,

ComCtrls;

const

IOCTL\_DISK\_GET\_DRIVE\_GEOMETRY = $70000;

type

// версия ОС

TWinVersion = (wvUnknown,wv95,wv98,wvME,wvNT3,wvNT4,wvW2K,wvXP,wv2003,wvVista,wv7);

//геометрия

TDiscGeometry = packed record

Cylinders: Int64; // колличество цилиндров

MediaType: DWORD; // тип носителя

TracksPerCylinder: DWORD; // дорожек на цилиндре

SectorsPerTrack: DWORD; // секторов на дорожке

BytesPerSector: DWORD; // байт в секторе

end;

//TForm

TForm3 = class(TForm)

Button1: TButton;

DriveComboBox1: TDriveComboBox;

Panel1: TPanel;

Button2: TButton;

Grid1: TStringGrid;

Button3: TButton;

GroupBox1: TGroupBox;

bpbList: TValueListEditor;

Label1: TLabel;

GroupBox2: TGroupBox;

Disks: TLabel;

BitBtn1: TBitBtn;

GroupBox3: TGroupBox;

Memo1: TMemo;

Grid2: TStringGrid;

Label2: TLabel;

Memo2: TMemo;

Label3: TLabel;

procedure Button1Click(Sender: TObject);

procedure DiscGeometryShow;

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure FormActivate(Sender: TObject); // Result = LoDWORD

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form3: TForm3;

hDrive,hMBRDrive: THandle;

DiscGeometry:TDiscGeometry;

typeOfDisc:string;

implementation

{$R \*.dfm}

// определение версии ОС

function DetectWinVersion : TWinVersion;

var

OSVersionInfo : TOSVersionInfo;

begin

Result := wvUnknown; // Неизвестная версия ОС

OSVersionInfo.dwOSVersionInfoSize := sizeof(TOSVersionInfo);

if GetVersionEx(OSVersionInfo)

then

begin

case OSVersionInfo.DwMajorVersion of

3: Result := wvNT3; // Windows NT 3

4: case OSVersionInfo.DwMinorVersion of

0: if OSVersionInfo.dwPlatformId = VER\_PLATFORM\_WIN32\_NT

then Result := wvNT4 // Windows NT 4

else Result := wv95; // Windows 95

10: Result := wv98; // Windows 98

90: Result := wvME; // Windows ME

end;

5: case OSVersionInfo.DwMinorVersion of

0: Result := wvW2K; // Windows 2000

1: Result := wvXP; // Windows XP

2: Result := wv2003; // Windows 2003

end;

6: case OSVersionInfo.DwMinorVersion of

0: Result := wvVista; // Windows Vista

1: Result := wv7; // Windows 7

end;

end;

end;

end;

// для вывода версии

function DetectWinVersionStr : string;

const

VersStr : array[TWinVersion] of string = (

'Unknown',

'Windows 95',

'Windows 98',

'Windows ME',

'Windows NT 3',

'Windows NT 4',

'Windows 2000',

'Windows XP',

'Windows 2003',

'Windows Vista',

'Windows Seven');

begin

Result := VersStr[DetectWinVersion];

end;

// для перемещения по диску

function \_\_Mul(a,b: DWORD; var HiDWORD: DWORD):DWORD;

asm

mul edx

mov [ecx],edx

end;

// Чтение сектора жесткго диска(Вызыв когда работа с диском начата)

function ReadSectors(hDrive:Thandle; StartingSector, SectorCount: DWORD;

Buffer: Pointer; BytesPerSector: DWORD): DWORD;

var

br,TmpLo,TmpHi: DWORD;

begin

Result := 0;

TmpLo := \_\_Mul(StartingSector,BytesPerSector,TmpHi);

if SetFilePointer(hDrive,TmpLo,@TmpHi,FILE\_BEGIN) = TmpLo then

begin

SectorCount := SectorCount\*BytesPerSector;

if ReadFile(hDrive,Buffer^,SectorCount,br,nil) then Result := br;

end;

end;

// вывод геометрии

procedure TForm3.DiscGeometryShow;

begin

Memo1.Clear;

Memo1.Text:='Вы просматриваете логический диск: '+DriveComboBox1.Drive+#13#10;

Memo1.Text:= memo1.Text + 'Количество цилиндров: '+inttoStr(DiscGeometry.Cylinders)+#13#10;

case DiscGeometry.MediaType of

12:typeOfDisc:=' жёсткий диск';

11:typeOfDisc:=' съёмный носитель';

end;

Memo1.Text:= memo1.Text + 'Тип носителя: '+typeOfDisc+#13#10;

Memo1.Text:= memo1.Text + 'Дорожек на цилиндре: '+intToStr(DiscGeometry.TracksPerCylinder)+#13#10;

Memo1.Text:= memo1.Text + 'Секторов на дорожке: '+intToStr(DiscGeometry.SectorsPerTrack)+#13#10;

Memo1.Text:= memo1.Text + 'Байт в секторе: '+intToStr(DiscGeometry.BytesPerSector)+#13#10;

end;

//создаем файл диска/выводим геомертию

//начинаем работу с жёстким диском

procedure TForm3.Button1Click(Sender: TObject);

var

discNameBPB,discNameMBR:string;

junk:Cardinal;

result:boolean;

begin

if hDrive <> 0 then CloseHandle(hDrive);

if hMBRDrive <> 0 then CloseHandle(hMBRDrive);

discNameBPB:='\\.\'+DriveComboBox1.Drive+':';

discNameMBR:='\\.\PHYSICALDRIVE'+intToStr(0);

hMBRDrive:= CreateFile(PChar(discNameMBR),GENERIC\_READ, FILE\_SHARE\_READ or FILE\_SHARE\_WRITE,nil,OPEN\_EXISTING,0,0);

hDrive:= CreateFile(PChar(discNameBPB),GENERIC\_READ, FILE\_SHARE\_READ or FILE\_SHARE\_WRITE,nil,OPEN\_EXISTING,0,0);

if (hDrive <> INVALID\_HANDLE\_VALUE) and (hMBRDrive <> INVALID\_HANDLE\_VALUE) then

begin

Result := DeviceIoControl(hDrive,IOCTL\_DISK\_GET\_DRIVE\_GEOMETRY,nil,0,

@DiscGeometry,SizeOf(TDiscGeometry),junk,nil) and (junk = SizeOf(TDiscGeometry));

DiscGeometryShow();

end

else

begin

ShowMessage('Не возможно создать дескрипторы носителя');

exit;

end;

end;

// Чтение и рашифровка BPB

procedure TForm3.Button2Click(Sender: TObject);

var

i,j,k:integer;

read:Cardinal;

s:string;

buffer: array[1..131072] of byte;

begin

if ReadSectors(hDrive,0,1,@buffer,DiscGeometry.BytesPerSector) = DiscGeometry.BytesPerSector then

begin

for i:= 1 to 16 do

Grid1.Cells[i,0]:=intToHex((i-1),1);

i:=1; j:=1; k:=1;

while k<=DiscGeometry.BytesPerSector do

begin

if i=1 then Grid1.Cells[0,j]:=IntToHex(((j-1)\*16+(i-1)),3);

Grid1.Cells[i,j]:=IntToHex(Buffer[k],2);

inc(k);

inc(i);

if i>16 then

begin

i:=1;

j:=j+1;

Grid1.RowCount:= Grid1.RowCount+1;

end;

end;

// расшифровка данных BPB

for i:= 1 to 3 do

s:=s+ intToHex(buffer[i],2);

bpbList.Cells[1,1]:= s + 'h';

s:='';

// чтение имени ОС

for i:= 4 to 4 + 7 do

s:=s+ chr(buffer[i]);

bpbList.Cells[1,2]:= s;

s:='';

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

bpbList.Cells[1,3]:=IntToStr(buffer[$C+1] shl 8 + buffer[$C]);

bpbList.Cells[1,4]:=IntToStr(buffer[$C + 2]);

bpbList.Cells[1,5]:=intToStr(buffer[$C + 4] shl 8 + buffer[$C+3]);

bpbList.Cells[1,6]:=intToStr(buffer[$C+5]);

bpbList.Cells[1,7]:=IntToStr(buffer[$F + 4] shl 8 + buffer[$F + 3]);

bpbList.Cells[1,8]:=intToStr(buffer[$F+6] shl 8 + buffer[$F + 5]);

bpbList.Cells[1,9]:=intToHex(buffer [22],2);

if bpbList.Cells[1,9] = 'F8' then bpbList.Cells[1,9]:='Системный носитель инФормации'

else if bpbList.Cells[1,9] = 'FDh' then bpbList.Cells[1,9]:=' накопитель - 2 стороны, 9 секторов'

else if bpbList.Cells[1,9] = 'F9h' then bpbList.Cells[1,9]:=' накопитель - 2 стороны, 9 секторов'

else if bpbList.Cells[1,9] = 'F0h' then bpbList.Cells[1,9]:=' накопитель - 2 стороны, 15 секторов';

bpbList.Cells[1,10]:=intToStr(buffer[$F+9] shl 8 + buffer[$F+8]);

bpbList.Cells[1,11]:=intToStr(buffer[$F+11] shl 8 + buffer[$F+10]);

bpbList.Cells[1,12]:=intToStr(buffer[$F+13] shl 8 + buffer[$F+12]);

bpbList.Cells[1,13]:=intToStr(buffer[$F+17] shl 32 + buffer[$f+16]+ buffer[$f+15]+buffer[$f+14]);

bpbList.Cells[1,14]:=intToStr(buffer[$F + 21] shl 24 + buffer[$F + 20] shl 16 + buffer[$F + 19] shl 8 + buffer[$F + 18]);

bpbList.Cells[1,15]:=intToStr(buffer[$C + 25]);

bpbList.Cells[1,16]:=intToStr(buffer[$F + 23]);

bpbList.Cells[1,17]:=intToStr(buffer[$F + 24]);

bpbList.Cells[1,18]:=intToHex((buffer[$F + 28] shl 24 + buffer[$F + 27] shl 16 + buffer[$F + 26] shl 8 + buffer[$F + 25]),8);

//-----------вывод метки -----------

s:='';

for i:=29 to 40 do

s:=s+chr(buffer[$F+i]);

bpbList.Cells[1,19]:= s;

s:='';

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

//-- чтение ID ОС-------------

for i:= 55 to 62 do

s:=s+ chr(buffer[i]);

bpbList.Cells[1,20]:= s;

s:='';

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

end

else

ShowMessage('Не удалось считать блок параметров Bios');

end;

// чтение данных из структуры DriveInfo

procedure TForm3.Button3Click(Sender: TObject);

var

i,j,k:integer;

read:Cardinal;

s:string;

buffer: array[1..131072] of byte;

begin

Memo2.Clear;

if ReadSectors(hMBRDrive,0,1,@buffer,DiscGeometry.BytesPerSector) = DiscGeometry.BytesPerSector then

begin

for i:= 1 to 16 do

Grid2.Cells[i,0]:=intToHex((i-1),1);

i:=1; j:=1; k:=1;

while k<=DiscGeometry.BytesPerSector do

begin

if i=1 then Grid2.Cells[0,j]:=IntToHex(((j-1)\*16+(i-1)),3);

Grid2.Cells[i,j]:=IntToHex(Buffer[k],2);

inc(k);

inc(i);

if i>16 then

begin

i:=1;

j:=j+1;

Grid2.RowCount:= Grid2.RowCount+1;

end;

end;

// Расшифровка PartitionTable

// первая часть

Memo2.Text:=Memo2.Text+'Первый элемент таблицы разделов системного диска: '+#13#10;

Memo2.Text:=Memo2.Text+'Признак загрузки: '+intToHex(buffer[$1bf],2)+#13#10;

Memo2.Text:=Memo2.Text+'Начало раздела диска{';

Memo2.Text:=Memo2.Text+'Головка: '+intToStr(buffer[448]);

Memo2.Text:=Memo2.Text+' Сектор: '+intToStr(buffer[449]);

Memo2.Text:=Memo2.Text+' Цилиндр: '+intToStr(buffer[450]);

Memo2.Text:=Memo2.Text+' }'+#13#10;

Memo2.Text:=Memo2.Text+'Тип раздела {';

Memo2.Text:=Memo2.Text+'Ос: '+intToStr(buffer[451]);

Memo2.Text:=Memo2.Text+' }'+#13#10;

Memo2.Text:=Memo2.Text+'Конец раздела диска{';

Memo2.Text:=Memo2.Text+'Головка: '+intToStr(buffer[$1c4]);

Memo2.Text:=Memo2.Text+' Сектор: '+intToStr(buffer[$1c5]);

Memo2.Text:=Memo2.Text+' Цилиндр: '+intToStr(buffer[$1c6]);

Memo2.Text:=Memo2.Text+' }'+#13#10;

Memo2.Text:=Memo2.Text+'Номер Сектора {';

Memo2.Text:=Memo2.Text+': '+intToStr(buffer[458]shl 32+buffer[457]+buffer[456]+buffer[455]);

Memo2.Text:=Memo2.Text+' }'+#13#10;

Memo2.Text:=Memo2.Text+'Размер раздела {';

Memo2.Text:=Memo2.Text+': '+intToStr(buffer[462]shl 24 +buffer[461] shl 16 +buffer[460]shl 8+buffer[459]);

Memo2.Text:=Memo2.Text+' }'+#13#10+#13#10+#13#10;

//вторая часть

Memo2.Text:=Memo2.Text+'Второй элемент таблицы разделов системного диска: '+#13#10;

Memo2.Text:=Memo2.Text+'Признак загрузки: '+intToHex(buffer[$1CF],2)+#13#10;

Memo2.Text:=Memo2.Text+'Начало раздела диска{';

Memo2.Text:=Memo2.Text+'Головка: '+intToStr(buffer[465]);

Memo2.Text:=Memo2.Text+' Сектор: '+intToStr(buffer[466]);

Memo2.Text:=Memo2.Text+' Цилиндр: '+intToStr(buffer[467]);

Memo2.Text:=Memo2.Text+' }'+#13#10;

Memo2.Text:=Memo2.Text+'Тип раздела {';

Memo2.Text:=Memo2.Text+'Ос: '+intToStr(buffer[468]);

Memo2.Text:=Memo2.Text+' }'+#13#10;

Memo2.Text:=Memo2.Text+'Конец раздела диска{';

Memo2.Text:=Memo2.Text+'Головка: '+intToStr(buffer[469]);

Memo2.Text:=Memo2.Text+' Сектор: '+intToStr(buffer[470]);

Memo2.Text:=Memo2.Text+' Цилиндр: '+intToStr(buffer[471]);

Memo2.Text:=Memo2.Text+' }'+#13#10;

Memo2.Text:=Memo2.Text+'Номер Сектора {';

Memo2.Text:=Memo2.Text+': '+intToStr(buffer[$1D7+3]shl 24+buffer[$1D7+2] shl 16+buffer[$1D7+1]shl 8 +buffer[$1D7]);

Memo2.Text:=Memo2.Text+' }'+#13#10;

Memo2.Text:=Memo2.Text+'Размер раздела {';

Memo2.Text:=Memo2.Text+': '+intToStr(buffer[$1db+3]shl 24 +buffer[$11db+2] shl 16 +buffer[$1db+1]shl 8+buffer[$1db]);

Memo2.Text:=Memo2.Text+' }'+#13#10+#13#10+#13#10;

end

else

ShowMessage('Не удалось считать PartitionTable');

end;

// получение инфо о нужных дисках

procedure TForm3.FormActivate(Sender: TObject);

var

S,SOut : String;

i:Integer;

VolumeName,FileSystemName,NewDiscNumber : String;

MaxComponentLength,FileSystemFlags:LongWord;

// от переменной VolumeSerialNo : DWord;

begin

S:=GetDisks(DiskHDD); {Получаем список Жёстких дисков (Параметр DiskHDD)}

s:=s+GetDisks(DiskFDD);

s:=s+GetDisks(DiskNet);

s:=s+GetDisks(DiskCDROM);

s:=s+GetDisks(DiskRAM);

SOut:='';

For i:=1 to Length(S) do {Получаем информацию о всех дисках и пишем в TLabel на форме}

begin

{Если диск существует/вставлен ...}

if GetHDDInfo(S[I], VolumeName, FileSystemName,NewDiscNumber,

MaxComponentLength, FileSystemFlags) then {... тогда собираем информацию}

SOut:=SOut+

'Диск: '+S[I]+#13#10+

'Метка: '+VolumeName+#13#10+

'Файловая система: '+FileSystemName+#13+#10+

'Серийный номер: '+NewDiscNumber+#13+#10+

'Макс. длина имени файла: '+IntToStr(MaxComponentLength)+#13+#10+

'Flags: '+IntToHex(FileSystemFlags,4)+#13#10+#13#10;

end;

Disks.Caption:=SOut; {Выводим в компонент TLabel полученные данные о дисках}

end;

end.

**5) Модуль ThrdRcvr (модуль описывающий работу потока – приёмника сообщений):**

unit ThrdRcvr; // принимающий поток

interface

uses

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

Dialogs,StdCtrls, Buttons, Menus, ComCtrls, ActnMan, ActnColorMaps;

type

TThreadReceiver = class(TThread)

private

PBaseAdress:PChar;

TextString:String;

nlineSize:integer;

protected

procedure Execute; override;

public

end;

implementation

uses

Unit1;

// Считываем соержимое из проекции файла в Memo1

procedure TThreadReceiver.Execute;

begin

while (not terminated) do

begin

WaitForSingleObject(Form1.HEvent, INFINITE);

ResetEvent(Form1.HEvent);

try

// резерв. память в потоке для этотой проекции

PBaseAdress:=MapViewOfFile(Form1.HFileSender, FILE\_MAP\_READ,0,0,65536);

if (PBaseAdress = nil) then

begin

CloseHandle(Form1.HFileSender);

st:='не удалось отобразить файл на адресное пространство';

Application.MessageBox(st,'Сообщение...',MB\_OK);

form1.BitBtn2.Visible:=false;

form1.Button1.Visible:=false;

exit;

end;

nlineSize:=integer(PBaseAdress^);

SetLength(TextString,nlineSize);

CopyMemory(PChar(TextString),Pointer(Integer(PBaseAdress)+4),nlineSize);

Form1.Memo1.Text:=Form1.Memo1.Text+'---------'+#13#10;

Form1.Memo1.Text:=Form1.Memo1.Text+TextString+#13#10;

sendmessage(Form1.memo1.handle, em\_linescroll, 0, Form1.memo1.lines.count-1);

UnmapViewOfFile(PBaseAdress);

finally

end;

end;

end;

end.

**6) Модуль HDDInfo (модуль получающий информацию о жёстких дисках):**

unit HDDInfo;

interface

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

Dialogs, StdCtrls, FileCtrl, ExtCtrls, Grids, ValEdit;

const {Константы для TypeOfDisk функции GetDisks}DiskUnknown=0; {Неизвестные диски}

DiskNone=1; {Отсутствующие диски}

DiskFDD=DRIVE\_REMOVABLE; {Съёмные диски, дискеты}

DiskHDD=DRIVE\_FIXED; {Не съёиные диски, жёсткие диски}

DiskNet=DRIVE\_REMOTE; {Сетевые диски}

DiskCDROM=DRIVE\_CDROM; {CD ROM}

DiskRAM=DRIVE\_RAMDISK; {Диски в ОЗУ}

function GetDisks(TypeOfDisk : Word) : String;

function GetHDDInfo(Disk : Char;Var VolumeName, FileSystemName,NewDiscNumber : String;

MaxComponentLength, FileSystemFlags:LongWord) : Boolean;

implementation

//получить имена доступных дисков

function GetDisks(TypeOfDisk : Word) : String;{Получить имена нужных дисков}

var

DriveArray : array[1..26] of Char;

i:integer;

beginDriveArray:='ABCDEFGHIJKLMNOPQRSTUVWXYZ';

for i:= 1 to 26 do

if GetDriveType(PChar(DriveArray[I]+':\')) = TypeOfDisk then

Result := Result+DriveArray[I];

end;

// определение правильного серийного номера диска

function GetReplaceCDNumber(num: String): String;

var

i, len: Integer;

begin

Result:= '';

len:= Length(num);

if len <> 8 then exit;

for i:= 1 to (len div 2) do begin

Dec(len);

Result:= Result + num[len];

Result:= Result + num[len+1];

Dec(len);

end;

end;

// получить инфо о дисках с нужными именами

function GetHDDInfo(Disk : Char;Var VolumeName, FileSystemName,NewDiscNumber : String;

MaxComponentLength, FileSystemFlags:LongWord) : Boolean;

var

\_VolumeName,\_FileSystemName:array [0..MAX\_PATH-1] of Char;

\_VolumeSerialNo,\_MaxComponentLength,\_FileSystemFlags:LongWord;

\_NewDiscNumber:string;

begin

if GetVolumeInformation(PChar(Disk+':\'),\_VolumeName,MAX\_PATH,@\_VolumeSerialNo,

\_MaxComponentLength,\_FileSystemFlags,\_FileSystemName,MAX\_PATH) then

begin

NewDiscNumber:=IntToHex(integer(\_VolumeSerialNo),8);

NewDiscNumber:=GetReplaceCDNumber(NewDiscNumber);

VolumeName:=\_VolumeName;

MaxComponentLength:=\_MaxComponentLength;

FileSystemFlags:=\_FileSystemFlags;

FileSystemName:=\_FileSystemName;

Result:=True;

end

else

Result:=False;

end;

end.

# Список литературы:

1) М. Фленов «Библия Delphi»

2) А. Танцер «Синхронизация процессов при работе с Windows»

3) Сайт CitForum > http://citforum.ru/programming/delphi/disk\_editor/ </

4)Cайт Delphi-Faq > http://delphi-faq.zoxt.net/a79.htm <.

5) Cайт DelphiSource.