Государственный научный центр Арктический и антарктический научно-исследовательский институт

Краткий отчёт по проекту в рамках договора №35/010 от 9 июля 2011 года «Реализация и внедрение методов обработки и хранения данных вертикального зондирования ионосферы»

подготовил:
Гришенцев А. Ю.
год 2011, октябрь 28

28 страниц
4 иллюстрации

Санкт-Петербург

Оглавление

1. Введение

В работе рассмотрено решение некоторых практических задач, в рамках договора №35/010 от 9 июля 2011 года, обработки и хранения, данных высотного зондирования ионосферы.

2. Цели и задачи

Цели:

Организация систематического хранения предварительная обработка и подготовка к последующему анализу данных с ионосферных станций, в рамках проекта по исследованию высокоширотной ионосферы с помощью метода вертикального зондирования сетью станций на базе аппаратно-программных комплексов CADI.

Задачи:

анализ форматов данных на соответствие документации, выявление недокументированных особенностей;

разработка программных библиотек чтения и подготовки к последующей обработке данных ионосферной станции высотного зондирования;

разработка программы визуализации и тестирования математических моделей для обработки данных высотного зондирования;

разработка структуры БД для хранения данных зондирования ионосферы; разработка скрипта (php, MySQL) разбора данных файлов ионограмм.

3. Разработка программных библиотек чтения и подготовки к последующей обработке данных ионосферной станции высотного зондирования

3.1. Анализ формата данных высотного зондирования ионосферы

Анализ производился на основе разбора реальных данных и спецификации формата [1]. Были выявлены неоднозначности и несоответствия документации и реальных данных.

Далее приведено описание формата бинарных файлов (с расширениями *.md1, *.md2, *.md3, *.md4), упор сделан на спецификацию формата *.md4, т.к. именно данный формат используется в практических исследованиях части, которых посвящена данная работа.

Имена файлов

Все файлы CADI имеют специфическое имя содержащие информацию о дате и времени и тепе файла. Длинна имени файла 8 символов обозначающих дату и время, плюс точка и 3-х символьное расширение. В

общей сложности 12 символов. Имена файлов задаются в момент первого открытия, т.е. в момент создания.

Пример чтения даты времени из имени файла

Пример для имени файла 2A120417.md4:

2 – 2002 год;

A – январь (A, B, C, D, E … месяцы);

12 – число;

04 – часа;

17 – минут.

Типы расширений

Типы расширений:

md1 - высота 510 км, одна или несколько "фиксированных" частот сканирования;

md2 - высота диапазоне 510 км, ионограмма по диапазону частот;

md3 - то же, что md1 высота 1020 км;

md4 - то же, что md2 высота 1020 км.

Обратим внимание, что в некоторых полученных файлах (.md4), было выявлено расхождение даты содержащейся в имени файла и внутри файла. Поэтому при практической работе, в качестве базовой даты, было выбрано значение (timestamp*) содержащиеся в файле и соответствующее по описанию [1] моменту создания файла.*

Структура файлов

Пример ручной дешифрации файла .md4

47524B204E6F7620 20352030313A3231 3A33362032303130 0A // отметка времени (timestamp)

49 // тип файла, значения: 49 or 48 («H» или «I» – почасовой или индивидуальный) используется «I»

9001 // 90 + 01*256 = 400 (no of freq) число частот для зондирования

08 // длина доплеровской серии (используется при наклонном зондировании)

5A00 // минимальная высота 5A + 00 * 256 = 90 km

FC03 // максимальная высота FC + 03 * 256 = 1020 km

14 // импульсов в секунду

02 // число импульсов усреднённое значение (no of pulses averaged)

---

* timestamp – стандартное название формата применяемое в ОС Linux, Unix некоторых базах данных, при этом под одним названием понимаются несколько различные спецификации, например: в Linux и MySQ.

9001 // базовый порог 100 = 100 × мин мощность (биты кв), перед сохранение

// (basethreshold100 = 100× min power (bits sq.) before save)

5000 // шумовой порог 100 = 100 × множитель для средней мощности шума

// noisethreshold100 = 100×multiplier for average noise power

//(bits sq.) to set threshold based on noise

01 // минимальное количество dopplers перед сохранением (minimum number of dopplers before saving;), используется для наклонного зондирования;

D007 // число секунд между образцами записи

41 // усиление (коэфициент) или 'N'= 0x4E; (gaincontrol: 'N'= none, or a number) (в символах '0'...'7')

46 // признак фильтрации FFT

// 'F' = делать FFT (затем сохранить все компоненты выше порога),

// 'T' = timeseq (БПФ, весь спектр, если какой-либо компонент выше порога)

// Или 'N' = нет (за исключением необработанных данных)

// sigprocess: 'F'=do FFT (then save all components above threshold),

// 'T'= timeseq (FFT, saves entire spectrum if any component is above threshold)

// or 'N'= none (save raw data)

01 // число приёмников (number of receivers;), используется – 1;

010000 0000000000000000 // резерв (11 байт)

// далее список частот (для этого файла 400 шт.) в формате float (по 4 байта на значение).

0024744900B57C49…

В связи с отсутствием подробного описания некоторых величин их связи с реальной обработкой данных требуются дополнительные исследования и уточнения у производителя для величин:

– basethreshold100;

– noisethreshold100;

– gaincontrol;

– FFT.

Для решения данного вопроса был направлен запрос к производителю ионосферных зондов CADI.

3.2. Разработка программных библиотек на языке C++

На основе произведённого анализа была разработана библиотека чтения файлов (*.md4), в виде класса. Особенностями и дополнительными возможностями данной библиотеки является:

– проверка файла на корректность по следующим признакам: минимальная длинна, допустимые значения;

– генерация и обработка исключительных ситуаций;

– соответствие стандартам языка C++ (*ISO/IEC 14882:2003*).

Программная реализация класса с подробными комментариями приведена в приложении.

3.3. Разработка библиотеки чтения бинарных данных на языке php

Для реализации операции чтения и предварительной обработки файлов (*.md4) на сервере, и последующем сохранением в архиве и базе данных был разработан скриптовый вариант программы чтения файлов в формате (*.md4) на языке php.

Реализация на языке php позволяет:

– выполнять операцию чтения файлов без перекомпилирования программы в различных операционных системах;

– производить оперативные изменения;

– относительно легко связываться с различными базами данных, в том числе MySQL.

Программная реализация класса с подробными комментариями приведена в приложении.

4. Разработка программы визуализации и тестирования математических моделей для обработки данных высотного зондирования

4.1. Разработка программы «SkySpectrum»

Для тестирования и визуализации данных бинарных файлов ионограмм с помощью разработанных программных библиотек была реализована программа «SkySpectrum», в виде Windows MDI приложения на языке C++ в среде IDE Embarcadero C++Builder (лицензионная).

Ниже приведён снимок экрана (рис. 2.1) при работе программы «SkySpectrum».

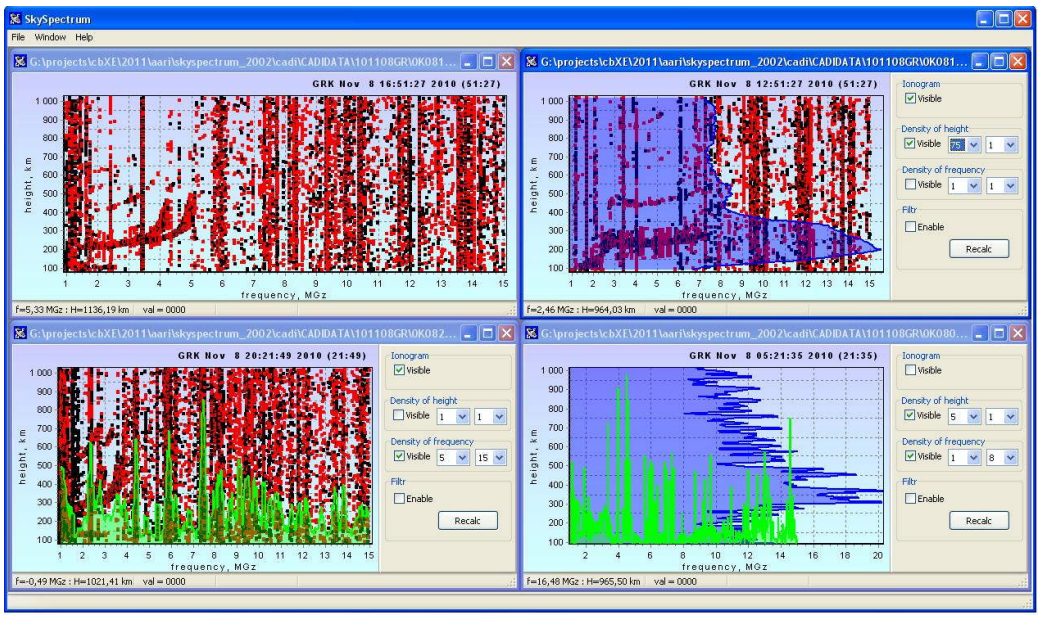


Рисунок 4.1. Окна программы визуализации бинарных ионограмм.

4.2. Краткая спецификация программы «SkySpectrum»

Программа «SkySpectrum» предназначена для чтения и визуализации, бинарных ионограмм высотного зондирования ионосферы, а в дальнейшем как платформа для тестирования и корректировки математических методов обработки и анализа бинарных файлов.

Возможности программы на текущий момент:

– открытие одного и множества бинарных файлов ионограмм (*.md4), с проверкой на корректность по ряду признаков;

– визуализация и масштабирование ионограмм;

– вывод численных координат положения курсора и амплитуды отражённого следа в текущей точке (в координатах ионограммы: высота, км; частота МГц; амплитуда отн. ед.);

– построение графиков плотности отражённых сигналов по частоте и высоте.

Минимальные системные требования:

– процессор PentiumIII или выше;

– ОЗУ от 250 мб;

– ОС Windows серии NT, поддержка графических режимов с применением DirectX.

Общее число строк программы 917274, из них только для «SkySpectrum» 2643. Объем исполнимого файла: 2.23 мб.

5. Разработка структуры БД для хранения данных зондирования ионосферы. Разработка скрипта (php, MySQL) разбора данных файлов ионограмм

5.1. Разработка структуры БД

Для построения структуры хранения данных в БД и возможности использования реляционных запросов необходимо учесть следующие принципы:

– соответствие таблиц, как образов, реальным физическим объектам, как прообразам;

– достаточное количество полей в таблицах для оперативного извлечения информации о физическом прообразе и его однозначной идентификации;

– устойчивость к программным и аппаратным сбоям и ошибкам;

– быстродействие, достижимое за счёт оптимизации запросов и хранимой информации.

В структуре БД предполагается выделить следующие кластеры (рис. 4.1):

– «Исходные данные»;

– «Обработка и анализ»;

– «Выводы, заключения, публикации, комментарии»;

– «Прогнозы».

Рассмотрим назначение каждого кластера в отдельности:

«Исходные данные» – данный кластер содержит в себе исходные данные, получаемые при высотном зондировании ионосферы (*.md4), а так же самой первичной обработки «на местах» получаемой в виде телеграмм IONKA.

«Обработка и анализ» – результаты обработки исходных данных машинной и человеком.

«Выводы, заключения, публикации, комментарии» – данный кластер содержит систематизированные, возможно обобщённые, результаты анализа (данных из кластеров «Исходные данные» и «Обработка и анализ»). В данном кластере могут быть размещены статьи, отчёты и прочие публикации. Реализован данный кластер может быть в виде корпоративной базы знаний включающий многолетний опыт и новые наработки в области исследования ионосферы.

«Прогнозы» – данный кластер является результатом работы математических моделей (интегрированных с БД) и позволяет: производить упорядоченное хранение данных прогнозов; оценку адекватности прогнозирования с последующей программной корректировкой.

Рисунок 5.1. Кластерная структура БД.

Как уже было отмечено выше, в представленной структуре БД является интегрированной, т.е. она взаимодействует с математическими моделями (как алгоритмическими объектами), пользователями, обслуживающим персоналом с помощью SQL запросов.

В данной работе рассматривается разработка программной поддержки, и структуры таблиц кластера «Исходные данные», поэтому далее преимущественно относится к этому кластеру.

5.2. Снижение нагрузки на БД

В связи с предполагаемым большим объёмом исходных данных (размер одного бинарного файла *.md4 25 – 50 кб), необходимо предусмотреть уменьшение объёмов таблиц путём разделения на «табличные» и «файловые» данные (рис. 4.2). К «табличным» данным отнесём: настройки оборудования, время и особенности регистрации ионограммы – эти данные будем хранить в таблицах; к «файловым» данным отнесём оцифрованные результаты регистрации отражённых от ионосферы сигналов в виде бинарных последовательностей.

Такой подход позволит значительно снизить нагрузку на БД и ускорить её работу.

Рисунок 5.2. Разделение данных.

5.3. Безопасность и целостность данных

С целью безопасности данных необходимо предусмотреть:

– все обращения к БД MySQL только через localhost, внешние запросы только через http (ftp) протокол при помощи «прокси-скриптов» (php, shell);

– ежедневное копирование БД;

– введение Hash полей в таблицы контролирующих целостность связанных файлов.

5.4. Таблицы

«ionobin» – таблица содержит служебную информацию о бинарных файлах ионограмм;

«ionotext» – таблица содержит информацию, присылаемую в текстовом виде ИОНКА;

«ionobinerror» – сообщения об ошибочных файлах *.md4.

Структуру таблиц «Ionobin» и «ionobinerror» можно найти в приложении.

Предполагается, что в начале каждого года таблицы за прошлый год переносятся в архив, к ним можно обращаться по именам «ionobinXXXX» и «ionotextXXXX», где XXXX, соответствует году в течении которого в данную таблицу производилась запись. Такой подход уменьшит объём файлов БД и снизит вероятность повреждения таблиц, а так же увеличит быстродействие. Данный пункт можно рассматривать как рекомендацию т.к. современные БД (в частности MySQL) достаточно быстро работают с большими таблицами.

5.5. Файлы

Файлы (*.md4) хранятся в директориях построенных по следующему принципу:

/arhive/cadi/xxx/year/mmdd/ – ионограммы *.md4;

/arhive/cadi/errorfile/ – ошибочные файлы.

xxx – трёхбуквенное сокращение названия станции зондирования;

year – год регистрации ионограммы;

mmdd – месяц (два символа от 01 до 12) и день (два символа от 01 до 31) регистрации ионограммы.

Путь к любому файлу легко определить по соответствующей записи в таблице и, наоборот, по файлу легко найти соответствующую запись в таблице.

5.6. Разработка комплекса программ архивации бинарных данных в БД

Для архивации бинарных данных, получаемых в виде файлов *.md4, был разработан скрипт (комплекс программ) на языке php. При разработке скрипта использованы объектно-ориентированные возможности языка php5.

Такой подход позволил создать полноценный, платформонезависимый комплекс библиотек, которые можно использовать в дальнейших работах.

Среди разработанных классов стоит отметить:

MIArhiveMD4 – класс управления логикой программы архивации ионограмм;

MIAnalysisMD4 – класс извлечения «табличных» данных из файлов *.md4;

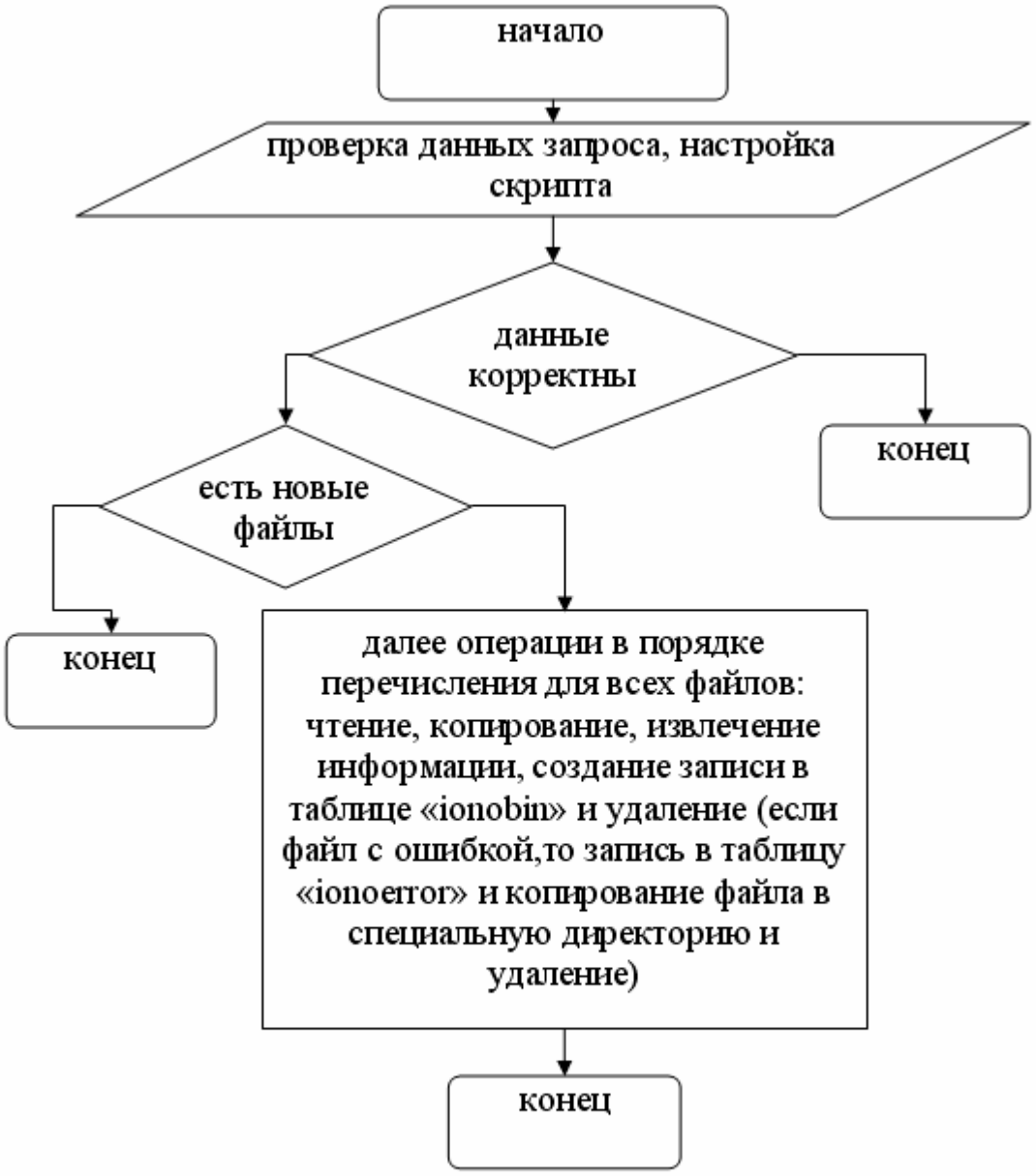
MISQLquery – класс автоматизации пакетных и одиночных запросов к БД MySQL;

MIFileOperation – класс работы с множеством файлов и директорий;

Дополнительно подключается файл MIBasic.php, содержащий набор некоторых базовых функций.

5.7. Алгоритм работы комплекса программ архивации бинарных данных

Далее приведён алгоритм (рис. 4.3) подробное описание управляющей структуры можно найти в приложении и исходных файлах программы.



Примечания:
– скрипт запускается на сервере по расписанию с периодом раз в две минуты;
– сообщения о всех ошибках регистрируются в «лог»-файл.

Рисунок 5.3. Упрощённый алгоритм работы комплекса программ архивации данных *.md4.

6. Список литературы

1. CADI. Sistem manuals. Scientific instrumentation limited. Canada/ Web: www.sil.sk.ca
2. Григин И.Е. PHP 5.1. Руководство разработчика (+CD). – СПб.: Питер, 2006. – 490 с.: ил.
3. Шилдт Г. Полный справочник по C. 4-е издание: перераб., и доп. Пер. с англ. – М.: издательский дом «Вильямс», 2007. – 704 с.: ил.

4. Страуструп Б. Язык программирования C++. Специальное издание. Пер. с англ. – М.: ООО «Бином-Пресс», 2008 г. – 1104 с.: ил.
5. Висвани В. Полный справочник по MySQL. Пер. с англ. – М.: издательский дом «Вильямс», 2006. – 528 с.: ил.
6. Харт, Джонсон М. Системное программирование в среде Windows, 3-е издание. Пер. с англ. – М.: издательский дом «Вильямс», 2005. – 528 с.: ил.
7. Скиена С. Алгоритмы. Руководство по разработке. 2-е изд. – СПб.: БХВ-Петербург, 2011. – 720 с.: ил.
8. Седжевик Р. Алгоритмы на C++. Пер. с англ. – М.: издательский дом «Вильямс», 2011. – 1056 с.: ил.
9. Стаханов А. Linux. 2-е издание: перераб., и доп. – СПб.: БХВ-Петербург, 2007. – 944 с.: ил.
10. Таймэн Б. FreeBSD 6. Полное руководство. Пер. с англ. – М.: издательский дом «Вильямс», 2008. – 1056 с.: ил.

7. Приложения

7.1. Класс чтения бинарных файлов ионограмм, язык C++

```
// ================================================================================
// development: Grishentcev Alex
// класс чтения ионограмм в форматах *.mdX ( x = 1,2,3,4)
// июль 2011 ver 1.0.2.1
// ================================================================================
// --------------------------------------------------------------------------
#ifndef MICadiReaderH
#define MICadiReaderH
// --------------------------------------------------------------------------
// Подключение библиотек
#include <string.h>
// --------------------------------------------------------------------------
// макросы и определения перепроцессора
#define CADI_INT16(array,lowbyte) (array[lowbyte] + 256*array[lowbyte+1])
#define CADI_name_len       25  // длинна заголовка
#define CADI_char_len       1     // длинна char
#define CADI_int_len        2       // длинна int
#define CADI_float_len      4       // длинна float
#define CADI_res_len        11     // длинна резервного блока данных
#define CADI_listfreq_start  56    // начало списка частот (в массиве данных)
#define CADI_max_height     340     // максимальная высота
// (размер массива принятых отраж. сигналов)

// --------------------------------------------------------------------------
// предопределяем тип для отражённого сигнала для удобства в случае замены
typedef unsigned char CADIsignal;

// фирменные структуры от CADI (порядок следования элементов не как у CADI)
// структура данных из заголовка
// внимание!!! порядок следования элементов структуры не соответствует
// порядку расположения данных в файлах *.mdX
// оптимизировано под размещение данных в Windows struct
struct MI_datablock {
  char filetype; // тип файла 'I' - отдельный файл (Individual);
  // 'H' - регулярный ежечасный файл (Hour)
  unsigned char dopplerlen;    // длина доплеровской серии
  unsigned char npulse;        // число импульсов в секунду
```

```
  unsigned char npulse_aver; // среднее число импульсов в секунду
  // (? 0x00 соответствует 256)
  unsigned char mindopler;     // minimum number of dopplers before saving;
  char gaincontrol;                // усиление 'N'= none, or 'A'-auto or numb. 0...7
  // если число то усиление в Дб,
  // например, при 2 затухание = 2*5 Дб
  char sigprocess;                // sigprocess: 'F'=do FFT
  // (then save all components above threshold),
  // 'T'= timeseq (FFT, saves entire spectrum
  // if any component is above threshold)
  // or 'N'= none (save raw data)
  unsigned char nreceiver; // number of receivers

  unsigned __int16 noffreq;          // число частот
  unsigned __int16 minheight;      // минимальная высота
  unsigned __int16 maxheight;     // максимальная высота
  unsigned __int16 basethreshold100; // basethreshold100 = 100× min power
  // (bits sq.) before save
  unsigned __int16 noisethreshold100; // noisethreshold100 = 100×multiplier
  // for average noise power
  unsigned __int16 secintervall;      // seconds between samples;
  unsigned char spares[CADI_res_len]; // резерв
};
// структура заголовка
struct MI_dataheader {
  char timestamp[CADI_name_len]; // символьная метка в 25 символов,
  MI_datablock dataBlock;        // struct
};
// структура контейнер для хранения параметров отражённых сигналов (по частотам)
struct MI_freqblock {
  unsigned char tmin;              // время минуты
  unsigned char tsec;              // время секунды
  unsigned char gainflag;        // флаг усиления
  unsigned char noiseflag;       // флаг шума
  unsigned __int16 nofrecord;        // номер записи
  unsigned __int16 avernoisepow; // средняя мощность шума
  // 10 (10 × averagenoisepower)
};

// ================================================================================ ==========
class MICadiReader {

  // ================================================================================ ==========
  // открытые свойства и методы
  public:
  // ---------------------------------------------------------------------------
  // Метод конструктор:
  // с инициализацией указателей, и переменных,
  // и установкой длинны массива данных (файла ионограммы)
  MICadiReader(int len);
  // ---------------------------------------------------------------------------
  // Метод деструктор
  // производит очистку памяти
  ~MICadiReader();
  // ---------------------------------------------------------------------------
  // Получаем длинну массива данных (файла ионограммы)
  int getLenData();
  // ---------------------------------------------------------------------------
  // Возвращает строку ошибки
  char* getError();
  // ---------------------------------------------------------------------------
  // Метод читатель заголовка с косвенной проверкой данных
  // Параметры:
  // указатель на массив данных
  // Возвращает:
  // если все в порядке: false
```

```
// в случае ошибки bool: true
bool readHeader(const unsigned char* data);
// ---------------------------------------------------------------------------
// Метод читатель списка частот
// Параметры:
// указатель на массив данных
// Возвращает:
// если все в порядке: false
// в случае ошибки bool: true
bool readFrequency(const unsigned char* data);
// ---------------------------------------------------------------------------
// Метод получает частоту по порядку номеров от 0..header->dataBlock.noffreq - 1
// Параметры:
// порядковый номер
// Возвращает:
// значение частоты (в случае выхода за предел диапазона номеров или если
// массив частот инициализирован
// возвращает 0 и заносит сообщение об ошибке)
float getFrequency(int numb);
// ---------------------------------------------------------------------------
// Метод читатель отражённых сигналов
// Параметры:
// порядковый номер частоты и высоты
// соотношение высот и индексов массивов (3*индекс) = (высота) км
// Возвращает:
// значение отражённого сигнала, при выходе за диапазон ноль
CADIsignal getReflSignal(unsigned __int16 nfreq, unsigned __int16 nheight);
// ---------------------------------------------------------------------------
// Метод читатель отражённых сигналов
// Параметры:
// указатель на массив данных
// Возвращает:
// если все в порядке: false
// в случае ошибки bool: true
bool readReflection(const unsigned char* data);
// ---------------------------------------------------------------------------
// Метод читатель отражённых сигналов
// Параметры:
// указатель на массив данных
// Возвращает:
// если все в порядке: false
// в случае ошибки bool: true
bool copyReflection(CADIsignal** array);
// ---------------------------------------------------------------------------
// заголовок данных
MI_dataheader* header;
// ---------------------------------------------------------------------------
// параметры отражённых сигналов (по частотам)
MI_freqblock* freqhead;
// ---------------------------------------------------------------------------
// ===========================================================================
// защищённые свойства и методы
// ---------------------------------------------------------------------------
// ---------------------------------------------------------------------------
// ===========================================================================
// закрытые свойства и методы
// ---------------------------------------------------------------------------
private:
// ---------------------------------------------------------------------------
// Строка сообщения об ошибке
char strError[256];
// ---------------------------------------------------------------------------
// длинна массива данных (файла ионограммы)
int datalen;
// ---------------------------------------------------------------------------
// список частот
```

```
  float* freqlist;
  // ---------------------------------------------------------------------------
  // указатель на двумерный массив отражённых сигналов
  // reflect_signal[i][j]  i - по частотам ; j - по высоте.
  // i = 0 ...noffreq - 1; j = 0 ... CADI_max_height - 1
  CADIsignal** reflect_signal;
  // ---------------------------------------------------------------------------
  // ===========================================================================
};
// ---------------------------------------------------------------------------
#endif


// ===========================================================================
// development: Grishentcev Alex
// tigerpost@ya.ru
// класс чтения ионограмм в форматах *.mdX ( x = 1,2,3,4)
// июль 2011 ver 1.0.2.1
// ===========================================================================
// ---------------------------------------------------------------------------
#pragma hdrstop
#include "MICadiReader.h"
// ---------------------------------------------------------------------------
#pragma package(smart_init)
// ===========================================================================

// ===========================================================================
// методы класса
// ---------------------------------------------------------------------------
// Метод конструктор
MICadiReader::MICadiReader(int len) {
  // начальные значения указателей
  header = 0;           // заголовок
  freqlist = 0;          // список частот
  freqhead = 0;        // параметры отражённых сигналов (по частотам)
  reflect_signal = 0; // отражённый сигнал

  // инициализация переменных
  datalen = len; // длинна массива данных (файла ионограммы)
}
// ---------------------------------------------------------------------------
// Метод деструктор
MICadiReader::~MICadiReader() {
  // чистим строку ошибок
  strcpy(strError, "");
  try {
    // чистим память
    if (header)      // заголовок
       delete header;

    if (freqlist)     // список частот
       delete[] freqlist;

    if (freqhead) // параметры отражённых сигналов
       delete[] strError;

    if (reflect_signal) { // массив отражённых сигналов
      for (int i = 0; i < header->dataBlock.noffreq; i++)
       delete[] reflect_signal[i];
      delete[] reflect_signal;
    }

  } catch (...) {

  }
}
```

```
// ---------------------------------------------------------------------------
// Получаем длинну массива данных (файла ионограммы)
int MICadiReader::getLenData() {
  return (datalen);
}
// ---------------------------------------------------------------------------
// Возвращает строку ошибки
char* MICadiReader::getError() {
  return (strError);
}
// ---------------------------------------------------------------------------
// Читатель заголовка с косвенной проверкой данных
bool MICadiReader::readHeader(const unsigned char* data) {
  // чистим строку ошибок
  strcpy(strError, "");
  // отлов исключений
  try {
    // если массив данных мала (возможно не установленна длинна)
    if (getLenData() < 56)
      throw("Data are very small");

    // если заголовок ещё не создан, выделяем память для заголовка
    if (!header)
      header = new MI_dataheader;
    else
      throw("Header is read");

    // последовательно копируем и форматируем данные
    memcpy(header->timestamp, data, CADI_name_len); // заголовок
    header->timestamp[CADI_name_len - 1] = 0;       // конец строки
    header->dataBlock.filetype = data[25];          // тип файла
    header->dataBlock.dopplerlen = data[28]; // длинна доплеровской серии
    header->dataBlock.npulse = data[33];      // число импульсов в секунду
    header->dataBlock.npulse_aver = data[34]; // число ипульсов для усреднения
    header->dataBlock.mindopler = data[39];
    // minimum number of dopplers before saving;
    header->dataBlock.gaincontrol = data[42]; // признак и значение усиления
    header->dataBlock.sigprocess = data[43];  // признак обработки FFT
    header->dataBlock.nreceiver = data[44];   // число приёмников
    header->dataBlock.noffreq = CADI_INT16(data, 26); // число частот
    header->dataBlock.minheight = CADI_INT16(data, 29); // минимальная высота
    header->dataBlock.maxheight = CADI_INT16(data, 31); // максимальная высота
    header->dataBlock.basethreshold100 = CADI_INT16(data, 35); // базовый порог
    header->dataBlock.noisethreshold100 = CADI_INT16(data, 37); // уровень шума
    header->dataBlock.secintervall = CADI_INT16(data, 40);
    // число секунд между записями
    memcpy(header->dataBlock.spares, & data[45], CADI_res_len);
    // резервный блок
    // далее осуществляем проверку коректности данных
    // по некоторым косвенным признакам
    // в случае некорректных данных генерируем сообщение об ошибке
    // проверяем timestamp
    for (int i = 0; i < CADI_name_len - 1; i++) {
      if (header->timestamp[i] < 0x20 || header->timestamp[i] > 0x7A)
        throw("Timestamp error");
    }
    // проверяем тип файла
    if (header->dataBlock.filetype != 'I' && header->dataBlock.filetype != 'H')
      throw("File type error");

    // минимальная высота
    if (header->dataBlock.minheight < 60)
      throw("Min height error");

    // число серий для усреднения (должно быть степень двойки)
    // 0x00 соответствует 256
```

```
    if ((header->dataBlock.npulse_aver - 1) & header->dataBlock.npulse_aver)
      throw("Number of pulses averaged error (2^N, N = 0...8)");

    // обработка FFT
    if (header->dataBlock.sigprocess != 'F' && header->dataBlock.sigprocess !=
      'N' && header->dataBlock.sigprocess != 'T')
      throw("FFT flag error");

    // усиление
    if ((header->dataBlock.gaincontrol < '0' || header->dataBlock.gaincontrol >
      '7') && header->dataBlock.gaincontrol != 'N' &&
      header->dataBlock.gaincontrol != 'A')
      throw("Gain control error");

    // обработка исключений
  } catch (const char* str) {
    strcpy(strError, str);
    return (true);
  } catch (...) {
    strcpy(strError, "Error reading header");
    return (true);
  }
  // выход в случае OK
  return (false);
}
// ---------------------------------------------------------------------------
// Метод читатель списка частот
bool MICadiReader::readFrequency(const unsigned char* data) {
  // чистим строку ошибок
  strcpy(strError, "");
  // если ошибки не было то выделяем память и
  // читаем список частот
  try {
    // если массив данных мала (возможно не установленна длинна)
    if (getLenData() < 56)
      throw("Data are very small");

    // если заголовок не прочтён
    if (!header)
      throw("Header is not read");

    // выделяем массив для нужного числа частот
    if (!freqlist)
      freqlist = new float[header->dataBlock.noffreq];
    else
      throw("Frequency is read");

    // копируем блок памяти (список частот)
    memcpy(freqlist, & data[CADI_listfreq_start],
      CADI_float_len* header->dataBlock.noffreq);

    // обработка исключений
  } catch (const char* str) {
    strcpy(strError, str);
    return (true);
  } catch (...) {
    strcpy(strError, "Error reading frequency");
    return (true);
  }
  return (false);
}
// ---------------------------------------------------------------------------
// Метод получает частоту по порядку номеров от 0...header->dataBlock.noffreq - 1
float MICadiReader::getFrequency(int numb) {
  // чистим строку ошибок
  strcpy(strError, "");
```

```
  try {
    // если массив freqlist не инициализирован
    if (freqlist == 0 || header == 0)
      throw("Header or frequency not read");

    // если за пределы диапазона
    if (numb < 0 || numb > header->dataBlock.noffreq - 1)
      throw("Out of range");

    // возвращаем результат
    return (freqlist[numb]);

    // обработка исключений
  } catch (const char* str) {
    strcpy(strError, str);
    return (0.0);
  } catch (...) {
    strcpy(strError, "Unknown error");
    return (0.0);
  }
}
// ---------------------------------------------------------------------------
// Метод читатель отражённых сигналов из данного класса
CADIsignal MICadiReader::getReflSignal(unsigned __int16 nfreq,
  unsigned __int16 nheight) {

  // чистим строку ошибок
  strcpy(strError, "");
  try {
    // если массив freqlist не инициализирован
    if (!reflect_signal)
      throw("Reflection signal not read");

    // если за пределы диапазона
    if ((nfreq >= header->dataBlock.noffreq) || (nheight >= CADI_max_height))
      throw("Out of range");
    // возвращаем значение отражённого сигнала
    return (reflect_signal[nfreq][nheight]);

    // обработка исключений
  } catch (const char* str) {
    strcpy(strError, str);
    return (0);
  } catch (...) {
    strcpy(strError, "Error read reflection signal");
    return (0);
  }
}
// ---------------------------------------------------------------------------
// Метод читатель отражённых сигналов из массива данных
bool MICadiReader::readReflection(const unsigned char* data) {
  // чистим строку ошибок
  strcpy(strError, "");
  // отлов исключений
  try {
    // проверка данных
    if (freqlist == 0 || header == 0)
      throw("Header or frequency not read");

    // проверка минимально возможной длинны массива данных
    // для хранения в нём отражённых сигналов
    if (datalen < 71)
      throw("Array data is very small");

    // выделяем память для массива списка
    // параметров отражённых сигналов (по частотам)
```

```
if (!freqhead)
  freqhead = new MI_freqblock[header->dataBlock.noffreq];
else
  throw("Frequency`s headers is read");

// выделяем память
// под двумерный массив отражённых сигналов
if (!reflect_signal) {
  reflect_signal = new CADIsignal*[header->dataBlock.noffreq];
  for (int i = 0; i < header->dataBlock.noffreq; i++)
    reflect_signal[i] = new CADIsignal[CADI_max_height];
} else
  throw("Reflected signals is read");

// инициализация данных нулями
// сначала нулевой элемент массива структур
freqhead[0].tmin = 0;
freqhead[0].tsec = 0;
freqhead[0].gainflag = 0;
freqhead[0].noiseflag = 0;
freqhead[0].nofrecord = 0;
freqhead[0].avernoisepow = 0;

// начальный элемент(массив) массива отраж. сигналов
for (int i = 0; i < CADI_max_height; i++)
  reflect_signal[0][i] = 0;

// инициализируем нулями остальные
for (int i = 1; i < header->dataBlock.noffreq; i++) {
  memcpy(& freqhead[i], & freqhead[0], sizeof(MI_freqblock));
  memcpy(reflect_signal[i], reflect_signal[0],
    CADI_max_height* sizeof(CADIsignal));
}

// далее заполняем массив структур данными
// начальное значение счётчика массива data
int counter = CADI_listfreq_start +
  CADI_float_len * header->dataBlock.noffreq;
// начальное значение счётчика частот
int nfreq = 0;
// начальное значение счётчика записей
// int nrecord = 0; не используем собираем данные только по первой записи
// счётчик высоты
int nheigth;
// установка времени первой записи
freqhead[0].tmin = data[counter++];
freqhead[0].tsec = data[counter++];
// цикл почастотам
do {
  // время для всех последующих частот время копируем из нулевой
  freqhead[nfreq].tmin = freqhead[0].tmin;
  freqhead[nfreq].tmin = freqhead[0].tmin;
  freqhead[nfreq].nofrecord = 0;
  freqhead[nfreq].gainflag = data[counter++];
  freqhead[nfreq].noiseflag = data[counter++];
  freqhead[nfreq].avernoisepow = CADI_INT16(data, counter);
  counter += 2;
  // теперь цикл по высоте
  while (data[counter] < 0xC9) { // признак продолжения высот
    // расчёт высоты
    nheigth = (data[counter + 1] & 0x80) ? (data[counter] + 0xC8) :
      data[counter];
    // отражённый сигнал в массив
    reflect_signal[nfreq][nheigth] = data[counter + 3];

    // doplerbin[nfreq][nheigth]  = data[counter + 2];
```

```
        // reflect_re[nfreq][nheigth] = data[counter + 3];
        // reflect_im[nfreq][nheigth] = data[counter + 4];

        // переходим к следующеё высоте
        counter += (2 + ((int)(data[counter + 1]& 0x7F))* 3);
      }
    }
    while ((++nfreq) < header->dataBlock.noffreq);

    // обработка исключений
  } catch (const char* str) {
    strcpy(strError, str);
    return (0.0);
  } catch (...) {
    strcpy(strError, "Unknown error");
    return (true);
  }
  // выходим
  return (false);
}
// ---------------------------------------------------------------------------
// Метод копировщик в массив отражённых сигналов
bool MICadiReader::copyReflection(CADIsignal** array) {
  // чистим строку ошибок
  strcpy(strError, "");
  // включаем обработку исключений
  try {
    // копируем массив отражённых сигналов
    for (int i = 0; i < header->dataBlock.noffreq; i++) {
      memcpy(& array[i], & reflect_signal[i],
        CADI_max_height* sizeof(CADIsignal));
    }
    // обработка исключений
  } catch (...) {
    strcpy(strError, "Copy error");
    return (true);
  }
  // выход всё в порядке
  return (false);
}
// ---------------------------------------------------------------------------
```

7.2. Класс чтения бинарных файлов ионограмм, язык php

```
<?php
/*
   +----------------------------------------------------------------------+
   | PHP version 5
   +----------------------------------------------------------------------+
   класс разбора файлов CADI (*.md4)

   +----------------------------------------------------------------------+
   Author: Grishentcev Alex
   <tigerpost@yandex.ru>
   +----------------------------------------------------------------------+
*/
////////////////////////////////////////////////////////////////////////////
// подключение внешних модулей
include_once('MIBasic.php'); // библиотека базовых функций
////////////////////////////////////////////////////////////////////////////

class MIAnalysisMD4 {
  ////////////////////// ПЕРЕМЕННЫЕ И КОНСТАНТЫ ////////////////////////////
  //
  public $datamd4; // бинарный файл
  // служебные данные из файла *.md4 и информация о файле
```

```
public $infomd4;
// массив (ключи массива соотвествуют именам полей в таблице ionobin)

public $message;   // информационное сообщение
public $full_name; // полное имя файла

// ========================================================================
////////////// КОНСТРУКТОР И ДЕСТРУКТОР ///////////////////////////////////
// ========================================================================
// метод конструктор класса
public function __construct() {
  // начальные значения
  $this->cleaner();
}
// ========================================================================
// метод деструктор класса
public function __destruct() {
  // чистим память
  $this->cleaner();
}
// ========================================================================
////////////////////// ОТКРЫТЫЕ МЕТОДЫ ////////////////////////////////////
// ========================================================================
// метод чтения файлов (*.md4) и получения некоторой информации
// Параметры:
// $file_name - имя файла
// Возвращает:
// TRUE  - ошибка: если массив
// $this->infomd4 == NULL - значит файл не прочитан;
// если массив
// $this->infomd4 != NULL - значит файл прочитан но слишком короткий;
// FALSE - всё в порядке
// помещает информацию в массив $this->infomd4[...]
public function file_read($file_name) {
  // чистим память
  $this->cleaner();

  // полное имя файла
  $this->full_name = $file_name;

  // читаем файл
  if (($this->datamd4 = MI_fileread($this->full_name)) == = NULL) {
    $this->infomd4['message'] = 'file not read'; // файл не прочитан
    return (true);
  }

  // заполнение данных в массив
  $this->infomd4['length'] = strlen($this->datamd4);  // длинна файла
  $this->infomd4['md5'] = md5($this->datamd4);        // md5 -hesh
  $this->infomd4['file_name'] = basename($file_name); // имя файла (без пути)


  // извлекаем служебную информацию
  if ($this->extract_info()) {
    // если ошибка (длинна файла слишком мала)
    $this->infomd4['message'] = 'file is too small';
    return (true);
  } else {
    $this->infomd4['message'] = 'file read and retrieval information success';
  }

  // выходим
  return (false);
}
// ========================================================================
// метод извлечения служебных данных из файлов (*.md4)
```

```
// Параметры:
// нет
// Возвращает:
// TRUE  - ошибка
// FALSE - всё в порядке
// помещает информацию в массив $this->infomd4[...]
public function extract_info() {
  // если длинна данных слишком мала выходим (true - error)
  if ($this->infomd4['length'] < 45)
    return (true);

  // извлекаем время и обозначение станции
  $this->extract_data(substr($this->datamd4, 0, 24));

  // тип ионограммы 'I', 'N', есле неопределён 'undefined'
  $this->infomd4['type'] = substr($this->datamd4, 25, 1);
  if ($this->infomd4['type'] != 'I' && $this->infomd4['type'] != 'H')
    $this->infomd4['type'] = 'undefined';

  // число частот зондирования
  $this->infomd4['nofreq'] =
    (ord($this->datamd4[26]) + 256 * ord($this->datamd4[27]));

  // длинна доплеровской серии
  $this->infomd4['doppler'] = ord($this->datamd4[28]);

  // минимальная высота, км
  $this->infomd4['minheight'] =
    (ord($this->datamd4[29]) + 256 * ord($this->datamd4[30]));

  // максимальная высота, км
  $this->infomd4['maxheight'] =
    (ord($this->datamd4[31]) + 256 * ord($this->datamd4[32]));

  // число импульсов в секунду
  $this->infomd4['npulse'] = ord($this->datamd4[33]);

  // число импульсов усреднение
  $this->infomd4['npulse_aver'] = ord($this->datamd4[34]);

  // базовый порог 100*(мин.мощн.)
  $this->infomd4['basethreshold100'] =
    (ord($this->datamd4[35]) + 256 * ord($this->datamd4[36]));

  // шумовой порог 100*(множитель для сред. мощ. шума)
  $this->infomd4['noisethreshold100'] =
    (ord($this->datamd4[37]) + 256 * ord($this->datamd4[38]));

  // минимальное количество доплеровских серий
  $this->infomd4['minndopler'] = ord($this->datamd4[39]);

  // число секунд между образцами записи
  $this->infomd4['secbetween'] =
    (ord($this->datamd4[40]) + 256 * ord($this->datamd4[41]));

  // коэфициент усиления
  $this->infomd4['gaincontrol'] = $this->datamd4[42];
  if ($this->infomd4['gaincontrol'] != 'N' && $this->infomd4['gaincontrol']
    != 'A' && !is_numeric($this->infomd4['gaincontrol'])) {
    $this->infomd4['gaincontrol'] = 'undefined';
  }

  // признак фильтрации FFT
  $this->infomd4['fft'] = $this->datamd4[43];
  if ($this->infomd4['fft'] != 'N' && $this->infomd4['fft']
    != 'F' && $this->infomd4['fft'] != 'T') {
```

```
    $this->infomd4['fft'] = 'undefined';
  }

  // число приёмников
  $this->infomd4['nreceivers'] = ord($this->datamd4[44]);

  // контрольный вывод
  // echo substr( $this->datamd4, 0, 24).' len='.$this->infomd4['length'].
  // ' md5='.$this->infomd4['md5'].' stan='.$this->infomd4['station'].' di='.
  // $this->infomd4['date_iono'].' type='.$this->infomd4['type'].' nf='.
  // $this->infomd4['nofreq'].' dl='.$this->infomd4['doppler'].' minh='.
  // $this->infomd4['minheight'].' maxh='.$this->infomd4['maxheight'].
  // ' npl='.$this->infomd4['npulse'].' bth100='.
  // $this->infomd4['basethreshold100'].' nth100='.
  // $this->infomd4['noisethreshold100'].
  // ' mindopl='.$this->infomd4['minndopler'].
  // ' secbet='.$this->infomd4['secbetween'].' gcont='.
  // $this->infomd4['gaincontrol'].' fft='.$this->infomd4['fft'].
  // ' nrec='.$this->infomd4['nreceivers'].'<br>';

  // OK выходим
  return (false);
}
// ============================================================================
// метод формирует путь к файлу по данным массива $this->infomd4[...]
// Параметры:
//
// Возвращает:
// путь или NULL - в случае ошибки
public function path_file() {
  if (isset($this->infomd4['station']) && isset($this->infomd4['date_iono']))
  {
    return ($this->infomd4['station'].'/'.date('Y',
      strtotime($this->infomd4['date_iono'])).'/'.date('md',
      strtotime($this->infomd4['date_iono'])).'/');
  }
  // выход с ошибкой
  return (NULL);

}
// ============================================================================
////////////////// ЗАЩИЩЕННЫЕ МЕТОДЫ ////////////////////////////////
// ============================================================================
// метод
// Параметры:
//
// Возвращает:
//
// protected function  ProtectedShablon() {}
// ============================================================================
////////////////// ЗАКРЫТЫЕ МЕТОДЫ ////////////////////////////////
// ============================================================================
// метод преобразования данных из cadi_timestamp в timestamp LINUX
// Параметры:
// $cadi_timestamp - строка (GRK Nov  2 04:40:20 2010)
// Возвращает:
//
private function extract_data($cadi_timestamp) {
  // сокращение станции
  $this->infomd4['station'] = trim(substr($cadi_timestamp, 0, 3));
  $this->infomd4['date_iono'] =
    date('Y-m-d H:i:s', strtotime(substr($cadi_timestamp, 4, 20)));
  return;
}
// ============================================================================
// метод чистильщик памяти
```

```
  // Параметры:
  //
  // Возвращает:
  //
  private function cleaner() {
    // чистим память
    unset($this->infomd4);
    unset($this->datamd4);
    unset($this->message, $this->full_name);
    // начальные значения
    $this->infomd4 = NULL;
    $this->datamd4 = NULL;
    $this->message = NULL;
    $this->full_name = NULL;
    return;
  }
  // ====================================================================
}
?>
```

7.3. Структура таблицы «ionobin»

Таблица 6.3.1. Структура таблицы «ionobin»

Поле	Тип	Null	По умолчанию	Комментарии
id	int(10)	Нет		id уникальный номер записи (ионограммы)
file_name	varchar(16)	Нет		имя исходного файла (*.md4)
station	enum('GRK', 'undefined')	Нет	undefined	код или название станции
date_iono	timestamp	Нет	0000-00-00 00:00:00	дата из (заголовка) файла
date_insert	timestamp	Нет	CURRENT_TIMESTAMP	дата вставки записи в таблицу
type	enum('H', 'I', 'undefined')	Нет		тип файла I-отдельная, H-часовая множественная
nofreq	smallint(5)	Нет		numb. of freq. -число частот
doppler	tinyint(3)	Нет		длина доплеровской серии
minheight	smallint(5)	Нет		минимальная высота, км
maxheight	smallint(5)	Нет		максимальная высота, км
npulse	tinyint(3)	Нет		число импульсов в секунду
npulse_aver	tinyint(3)	Нет		число импульсов (усреднение)
basethreshold100	varchar(5)	Нет		базовый порог 100 = 100 × мин мощность
noisethreshold100	varchar(5)	Нет		шумовой порог 100 = 100 × множитель для средней мощности шума
minndopler	tinyint(3)	Нет		минимальное количество dopplers перед сохранением
secbetween	smallint(5)	Нет		число секунд между образцами записи
gaincontrol	enum('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'A', 'N', 'undefined')	Нет		усиление (коэффициент) ('0'...'7')
fft	enum('N', 'F', 'T',	Нет		признак преобразования Фурье

Поле	Тип	Null	По умолчанию	Комментарии
	'undefined')			
nreceivers	tinyint(3)	Нет		число приёмников
length	mediumint(7)	Нет		длина файла
md5	varchar(32)	Нет		md5 hash code контрольный код целостности файла
message	varchar(255)	Да	*NULL*	сообщение обработчика файлов
description	text	Да	*NULL*	комментарий, описание

Таблица 6.3.2. Индексы таблицы «ionobin»

Имя индекса	Тип	Уникальных элементов	Поле
PRIMARY	PRIMARY	415	id
file_name_station	UNIQUE	415	file_name station
file_name	INDEX	Нет	file_name
station	INDEX	Нет	station
date_iono	INDEX	Нет	date_iono
date_insert	INDEX	Нет	date_insert
md5	INDEX	Нет	md5

Структура таблицы

Таблица 6.4.1. Структура таблицы «ionoerror»

Поле	Тип	Null	По умолчанию	Комментарии
id	int(10)	Нет		id уникальный номер записи (ионограммы)
file_name	varchar(255)	Нет		имя исходного файла (*.md4)
date_insert	timestamp	Нет	CURRENT_TIMESTAMP	дата вставки записи в таблицу
length	mediumint(7)	Нет		длина файла
md5	varchar(32)	Нет		md5 hash code контрольный код целостности файла
message	varchar(255)	Да	*NULL*	сообщение обработчика файлов
description	text	Да	*NULL*	комментарий, описание

Таблица 6.4.2. Индексы таблицы «ionoerror»

Имя индекса	Тип	Уникальных элементов	Поле
PRIMARY	PRIMARY	14	id
file_name	UNIQUE	14	file_name

7.4. Управляющий класс архивации данных *.md4

```
<?php
/*
+----------------------------------------------------------------------+
| PHP version 5
+----------------------------------------------------------------------+
|
+----------------------------------------------------------------------+
|  управляющий модуль архивацией файлов CADI (*.md4)
|
+----------------------------------------------------------------------+
| Copyright (c) 2011
| Author: Grishentcev Alex <tigerpost@yandex.ru>
+----------------------------------------------------------------------+
*/
//////////////////////////////////////////////////////////////////////////
//  подключение внешних модулей
```

```
include_once('MIBasic.php');            // библиотека базовых функций
include_once('MISQLquery.php');         // базовый клас MySQL запросов
include_once('MIFileOperation.php');    // класс для работы с файлами
include_once('MIAnalysisMD4.php');      // класс разбора файлов CADI (*.md4)

////////////////////////////////////////////////////////////////////////
class MIArhiveMD4 {
  ////////////////////// ПЕРЕМЕННЫЕ И КОНСТАНТЫ ////////////////////////////

  //=====================================================================
  ////////////// КОНСТРУКТОР И ДЕСТРУКТОР ////////////////////////////////
  //=====================================================================
  // метод конструктор класса
  public function __construct() {

    // производим поиск файлов в указанном каталоге
    $file = new MIFileOperation();
    if( $file->path_scan($GLOBALS['mi_heap_md4'], 'md4') ) {
      // ошибка сканирования (выводим сообщение)
      MI_funcException('Path:'.$GLOBALS['mi_heap_md4'].' scan (E1010)');
    }

    // обработка найденных файлов по порядку
    if( count($file->file_list) > 0 ) {
      // если какие-то данные получены то производим обработку
      $processing = new MIAnalysisMD4();  // объект класса MIAnalysisMD4 обработки данных
      $mysql = new MISQLquery();          // объект класса MISQLquery

      foreach( $file->file_list as $key=>$value ) {

        // проход по всему массиву (списку) найденных файлов
        if( $processing->file_read($value) ) {
          // если возникла ошибка
          // уточняем её свойство
          if( $processing->infomd4 == NULL ) {
            // файл не прочитан
            MI_funcException('File: '.$value.' reading failed');
          }

          if( $processing->infomd4 != NULL ) {
            // файл прочитан но слишком короткий
            MI_funcException('File: '.$value.' too small');
            // копируем файл в temp
            if( MI_createfile($GLOBALS['mi_bank_md4'],
                      '/errorfile/'.date('YmdHi').'/',
                      $processing->infomd4['file_name'],
                      $processing->datamd4) === NULL ) {
              // ошибка
              MI_funcException('File: '.$value.' copy to error archive failed');
            } else {
              // удаляем исходный файл
              if( !unlink($value) ) {
                // если удаление не прошло
                MI_funcException('File: '.$value.' delete failed');
              }
            }
            // делаем запись в БД
            $processing->infomd4['file_name'] = $value;
            if( !$mysql->array_insert( 'ionobinerror', $processing->infomd4 ) ) {
              MI_funcException('File: '.$value.' INSERT SQL error');
            }


          }

        } else {
```

```
        // если ошибки не обнаруженно
        // формируем полное имя пути для сохранения файла
        $path =  $processing->path_file( );
        // запись файла в архив
        if( MI_createfile($GLOBALS['mi_bank_md4'], $path,
                    $processing->infomd4['file_name'],
          $processing->datamd4) === NULL ) {
          // ошибка
          MI_funcException('File: '.$value.' copy to archive failed');

        } else {
          // всё в порядке, копирование успешно
          // делаем запись в БД
          if( !$mysql->array_insert( 'ionobin', $processing->infomd4 ) ) {
            MI_funcException('File: '.$value.' INSERT SQL error');
          }

          // удаляем исходный файл
          if( !unlink($value) ) {
            // если удаление не прошло
            MI_funcException('File: '.$value.' delete failed');
          }

        }

      }

    }

  } else {
    // если файлов нет значит папка пуста здесь можно её очистить
    // сканируем директорий в поисках любых файлов
    $file->clear();   // чистим предыдущий поиск
    $file->path_scan($GLOBALS['mi_heap_md4'], '*');

    if( count($file->file_list) > 0 ) {
      // если нашли - удаляем всё
      foreach( $file->file_list as $value ) {
        // удаляем исходный файл
        if( !unlink($value) ) {
          // если удаление не прошло
          MI_funcException('File: '.$value.' delete failed');
        }
      }
    }
    if( count($file->path_list) > 0 ) {
      // теперь можно очистить каталоги
      foreach( $file->path_list as $value ) {
        // удаляем исходный файл
        if( !rmdir($value) ) {
          // если удаление не прошло
          MI_funcException('Dir: '.$value.' delete failed');
        }
      }
    }

  }
  // чистим память
  unset($processing, $mysql);
  // выход
  return;
}
//=========================================================================
// метод деструктор класса
public function __destruct() {
  //
```

```
    }
    //============================================================================
    ////////////////////// ОТКРЫТЫЕ МЕТОДЫ ////////////////////////////////////
    //============================================================================

    // метод
    // Параметры:
    //
    // Возвращает:

    //============================================================================
    ////////////////////// ЗАЩИЩЕННЫЕ МЕТОДЫ ////////////////////////////////////
    //============================================================================
    // метод
    // Параметры:
    //
    // Возвращает:
    //
//    protected function  ProtectedShablon() {}
    //============================================================================
    ////////////////////// ЗАКРЫТЫЕ МЕТОДЫ ////////////////////////////////////
    //============================================================================
    // метод
    // Параметры:
    //
    // Возвращает:
    //
//    private function  PrivateShablon() {}
    //============================================================================
}
?>
```