ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Иркутский государственный университет путей сообщения»

(ФГБОУ ВО ИрГУПС)

Факультет «Управление на транспорте и информационные технологии»

Кафедра «Информационные системы и защита информации»

Дисциплина «Командная разработка программных системы»

Лабораторная работа № 2

ЛР.430200.09.03.04.120.ПЗ

Выполнил: Проверил:

студент гр. ПИ.1-14-1 доцент кафры «ИСиЗИ»

Козянко В. А. Курганская О. В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018г. «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018г.

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

Иркутск 2018

**Задание для организационной работы:**

Распределить роли участников группы.

Составить перечень задач на весь проект каждому участнику.

Определить способы общения между участниками команды.

Определить стандарты оформления кода.

**Задание для решения:**

Дан XML-документ, в котором описаны оценки студентов по некоторым предметам. Необходимо сгенерировать из него документ json формата, описывающий студентов и их оценки по предметам.

Доп. условия: язык программирования Go и его средства обработки XML, промежуточное хранение – ini-файл, обработка ошибок и защитное программирование обязательны.

**Роли участников:**

Леонид – разработчик № 1.

Надежда – лидер.

Егор – тестировщик.

Виктор – проектировщик, разработчик № 2.

**Взаимозаменяемость:**

Если нет тестировщика, то его заменяет разработчик № 2.

Если нет разработчика №2, то его заменяет разработчик №1 или тестировщик.

Если нет проектировщика, то его заменяет разработчик №1 или тестировщик.

Если нет разработчика №1, то его заменяет лидер.

Если нет лидера, то его заменяет разработчик №2.

**Задачи:**

Изучение предметной области – все.

Создание алгоритма программы, составление блок-схем – разработчик № 2.

Написание программы и исправление с комментариями – разработчик № 1, разработчик № 2.

Разработчик № 1– написание процедуры реализует интерфейс работы пользователя и прописывает связь функций, написание процедуры конвертирующей xml-файл в ini, написание процедуры конвертирующей ini-файл в json.

Разработчик № 2- написание процедуры выдающей список имен файлов xml в конкретной папке, написание процедуры записывающей ошибки в файл, написание процедуры создающую список для дальнейшей конвертации, собирает программу.

Тестирование программы – тестировщик.

Контроль за исполнением, инспекция, жестокое наказание, сбор отчета – лидер.

**Способы общения между участниками**

Вконтакте, GitHub

**Функции программы:**

ListXmlFile – на вход подаётся путь к папке, на выходе – список имен файлов xml. Путь к папке проверяется на правильность написания, если обнаружена ошибка, то выдаем ошибку.

ConvertXmlToIni – на вход имя xml-файла, на выходе имя созданного ini-файла (его можно назвать так же как и xml). Если появится какая-то ошибка (любая), то выводим её.

ConvertIniToJson - на вход имя ini-файла, на выходе имя созданного json-файла (его можно назвать так же как и xml). Если появится какая-то ошибка.

Logs – на вход поступает описание ошибки и производится запись в файл и вывод в консоль.

main – реализует интерфейс работы пользователя и прописывает связь функций.

CreateListFiles - на вход поступают: строка с типом создаваемого списка файлов (“ini”/ ”json”), список файлов для конвертации (для создания ini отправляется список xml, для создания json – ini).

**Описание работы программы:**

1. На вход поступает путь к папке, где хранятся xml-файлы. (вводит пользователь).
2. Если в папке нет xml-файлов, то вывести ошибку об отсутствии файлов в папке, иначе получить список xml-файлов.
3. Считать xml-файл. (Вывод сообщения об ошибки считывания, ошибки в структуре программы, ошибки в названии параметров).
4. Записать структуру xml-файла в ini-файл.
5. Считать ini-файл. Вывод сообщения об ошибки считывания, ошибки в структуре программы, ошибки в названии параметров.
6. Записать структуру ini-файла в json.
7. Удалить промежуточный ini-файл.
8. Если не существует в директории с xml-файлами папки json, то создать её. Сохранить результат в папку json. (вывод ошибки при сохранении).
9. Проделать шаги 3-5 пока в папке не пройдём по всем файлам.
10. Вывод сообщения, что преобразование прошло.

Сообщение об ошибках будут выводиться в отдельный файл с указанием даты, временем, имени файла, где произошла ошибка и описанием ошибки.

**Алгоритм работы функции ListXmlFile.**

На вход поступает путь к папке с xml-файлами.

1. Создать срез строкового типа, для занесения имя найденных xml файлов.
2. Проверить является ли путь директорией.
3. Если путь не является директорией, то выдать об этом ошибку, завершить выполнение программы с кодом ошибки .
4. Считывание имени файла.
5. Определение расширения файла.
6. Если файл имеет расширение xml, то занести его в срез.
   1. Найти последнее вхождение точки в строку.
   2. Выделить подстроку с позиции вхождения точки до конца (точка в рассмотрение не включается).
   3. Полученная подстрока = “xml”?
7. Повторить пункты 4-6 пока не пройдем по всем папкам.
8. Вернуть срез.

Блок схема работы функции ListXmlFile продемонстрирована на рисунке 1.

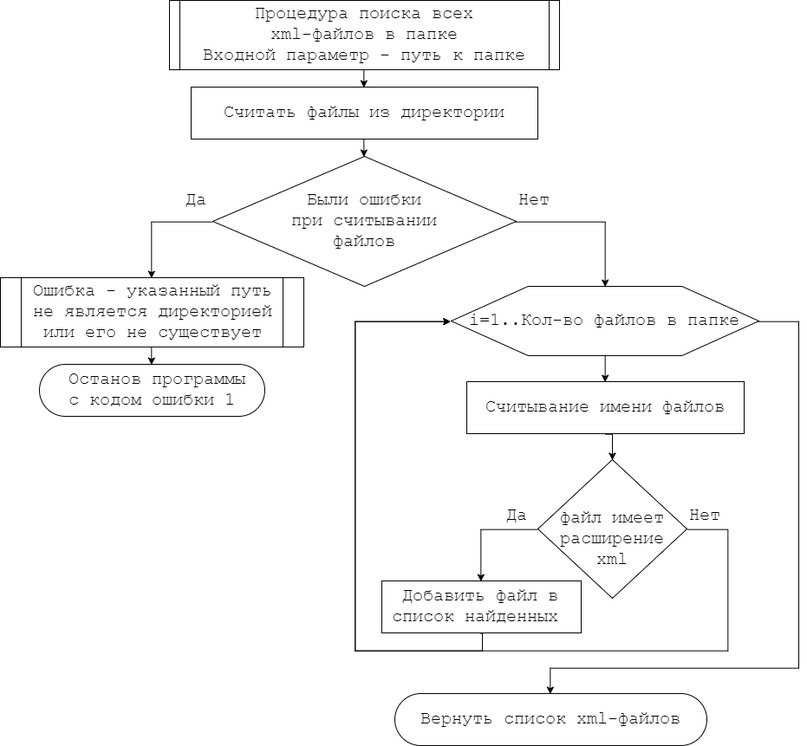


Рисунок 1 – Блок-схема работы функции ListXmlFile

**Алгоритм работы функции ConvertXmlToIni.**

На вход поступает имя xml-файла.

1. Попытка открытия файла.
2. Если при открытии файла возникли ошибки, то выдать ошибку, завершить выполнение функции.
3. Считывание файла xml в структуру xml-файла.
4. Запись данных в файл ini.
5. Если возникли ошибки вывести.
6. Вернуть имя ini-файла.

Блок схема работы функции ConvertXmlToIni продемонстрирована на рисунке 2.

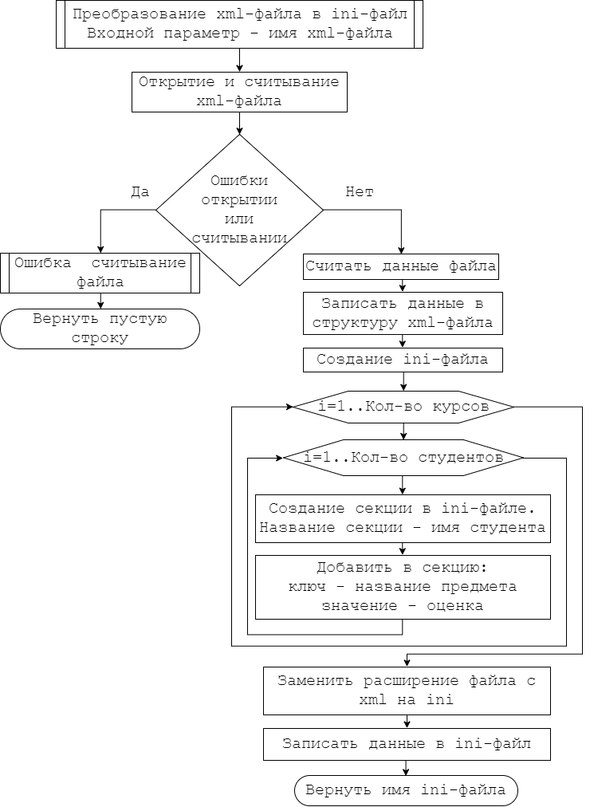


Рисунок 2 – Блок-схема работы функции ConvertXmlToIni

**Алгоритм работы функции ConvertIniToJson.**

На вход поступает имя ini-файла.

1. Попытка открытия файла.
2. Если при открытии файла возникли ошибки, то выдать ошибку, завершить выполнение функции.
3. Считать данные из ini.
4. Запись данные в json-файл.
5. Если возникли ошибки вывести.
6. Вернуть имя json-файла.

Блок схема работы функции ConvertIniToJson продемонстрирована на рисунке 3.

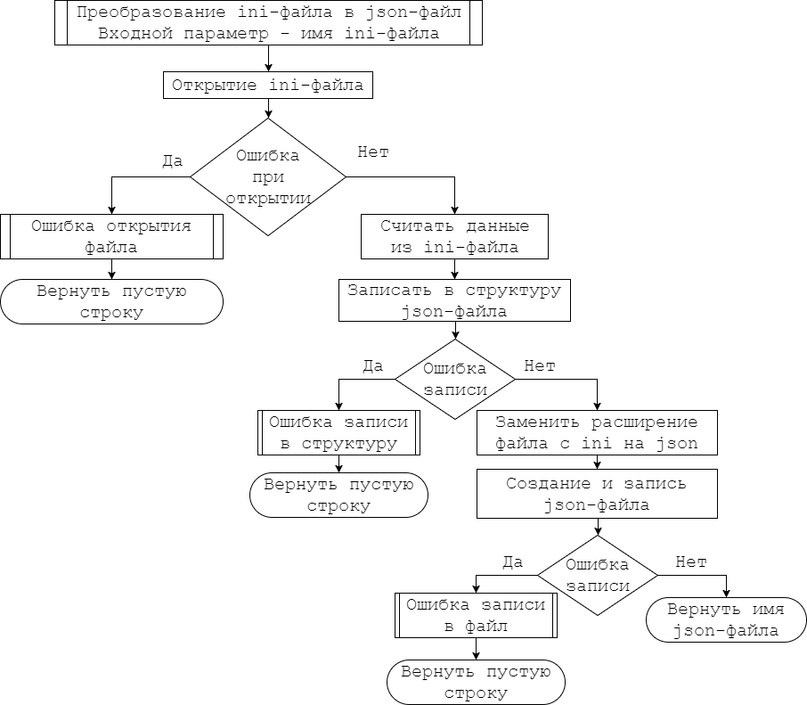


Рисунок 3 – Блок-схема работы функции ConvertIniToJson

**Алгоритм работы функции Logs.**

На вход поступает описание ошибки.

1. Попытка открыть файл для записи в него данных.
2. Если не получается, то создать новый файл.
3. Получить текущее время.
4. Вывести ошибку в консоль.
5. Записать в файл с ошибками дату и ошибку.
6. Закрыть файл.

Блок схема работы функции Logs продемонстрирована на рисунке 4.

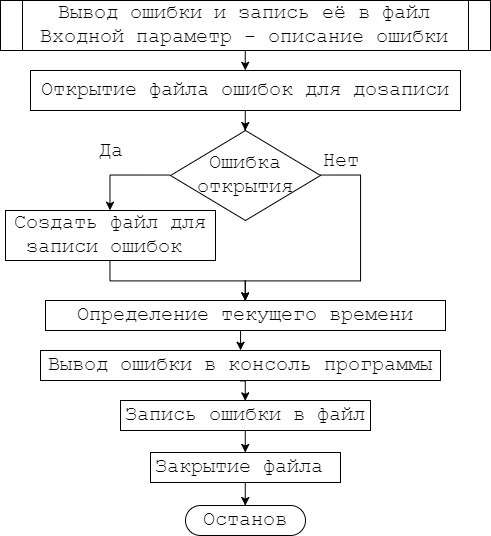


Рисунок 4 – Блок-схема работы функции Logs

**Алгоритм работы функции main.**

1. Ввести путь к папке с файлами xml.
2. Получение списка файлов xml с помощью функции **ListXmlFile.**
3. Если в папке отсутствуют файлы xml, то записать ошибку в файл, завершить выполнение программы.
4. Получить список ini-файлов, используя функцию **CreaeteListFile.**
5. Получить список json-файлов, используя функцию **CreaeteListFile.**
6. Вывести список успешно созданных json-файлов.

Блок схема работы функции main продемонстрирована на рисунке 5.

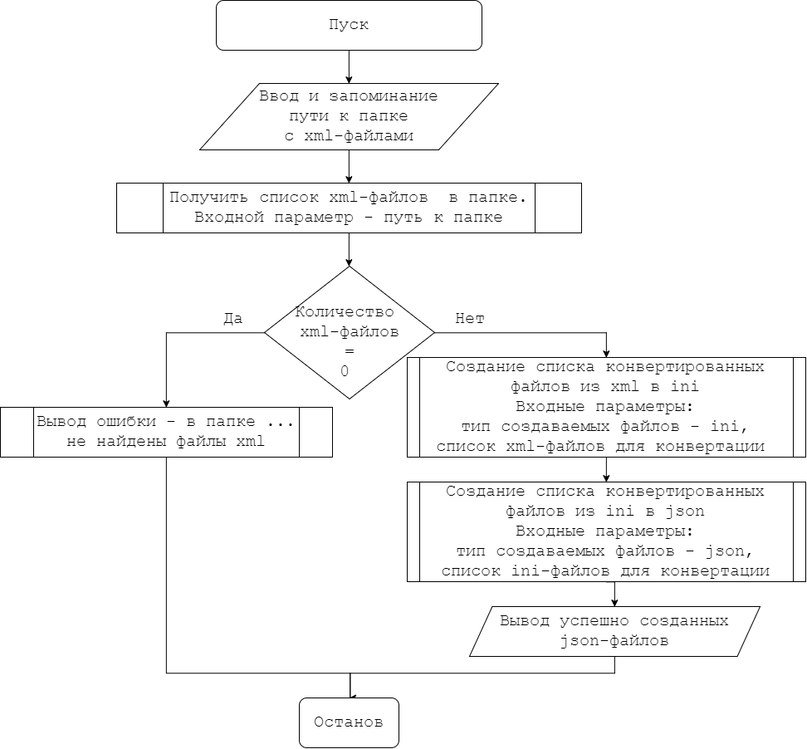


Рисунок 5 – Блок-схема работы функции main

**Алгоритм работы функции CreateListFiles.**

На вход поступают: строка с типом создаваемого списка файлов (“ini”/ ”json”), список файлов для конвертации (для создания ini отправляется список xml, для создания json – ini).

1) Создать переменную для определения функции конвертации: входные параметры – имя файла выходные – список файлов.

2) Если входной тип файлов = ini, то дальше будет использоваться функция **ConvertXmlToIni.**

3) Если входной тип файлов = json, то дальше будет использоваться функция **ConvertIniToJson.**

4) Входной параметр не принадлежит 2-м описанным типам, то выдача сообщения о неправильной передачи параметров в функцию, завершение программы c кодом ошибки 2.

5) Создать пустой список файлов.

6) Для каждого файла из списка выполнить функцию конвертации.

7) Если возвращенное значение не пустая строка, то занести имя файла в список.

8) Если список файлов пуст, то выдать ошибку об ошибки конвертации, завершить программу с кодом ошибки 3.

9) Вернуть список созданных файлов.

Блок схема работы функции CreateListFiles продемонстрирована на рисунке 6.

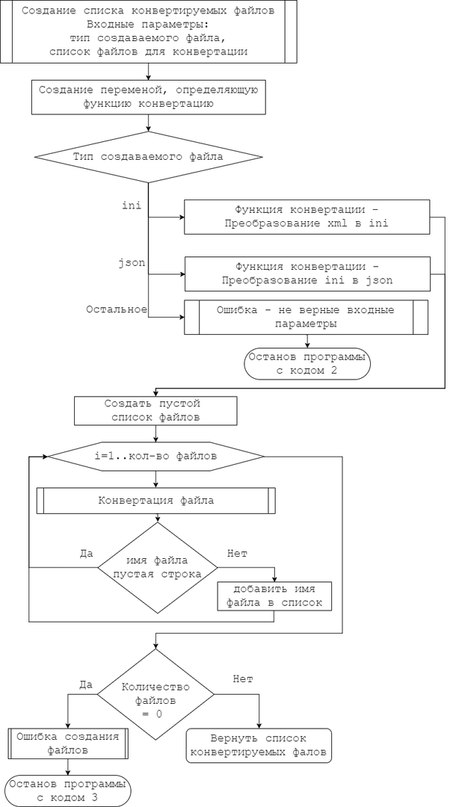


Рисунок 6 – Блок-схема работы функции CreateListFiles

**ListXmlFile()-**процедура поиска всех xml-файлов в папке.

Псевдокод

ЕСЛИ файлов нет ТО выдать ошибку

ЕСЛИ считаный файл xml ТО добавить его в список файлов

**ConvertXmlToIni()**-процедура преобразования xml-файла в ini-файл

Псевдокод

Считать структуру xml-файла

Записать структуру в ini-файл

**СonverIniToJson()**- процедура преобразования ini-файла в json-файл.

Псевдокод

Считать структуру ini-файла

Записать структуру в json-файл

**Log()**-процедура записи ошибки в файл.

Псевдокод

Создать файл ошибки

Вывести ошибку в консоль

Записать информацию об ошибке в файл

**CreateListFiles()** – процедура создания списка конвертированных файлов

**Структура входного файла xml**

<?xml version="1.0" encoding="UTF-8"?> - XML декларация

<courses> – Корневой элемент

<course name="название\_предмета"> – Элемент «Курс», с атрибутом название курса

<student name="Имя\_студента" mark="Оценка"/> – Элемент «Студент» с атрибутами «Имя» и «Оценка»

<student name=" Имя\_студента" mark="Оценка"/> – Элементов «Студент» может быть несколько

…

</course>

… - Элементов «Курс» с вложенными элементами «Студент» может быть несколько

</courses>

**Структура выходного файла json**

[

{

Students

{

“student” : “Имя\_студента”,

“course” : “Название\_предмета”,

“mark” : “Оценка за предмет”,

… – далее могут повторяться 2 предыдущие строки

},

{

…

}

}

]

**Структура промежуточного файла ini**

[Имя\_студента]

Название\_предмета : оценка

…

**Стандарты оформления кода**

Стиль написания имен переменных – camelCase.

Документационные комментарии – javaDoc .

Открывающая скобка логического блока на одном уровне с определением, закрывающая – на другой строке, исключения - } else {.

Отступ вложенных элементов в 2 пробела.

Математические и логические операторы выделяются пробелами.

Комментарии писать над строчкой с кодом.

Обязательное наличие псевдокода.

Обязательное наличие обработчика ошибок.

Обязательное наличие защитного программирования.

Обработку ошибок пишет разработчик, ошибка передаётся вверх по стеку вызова.

**Индивидуальная часть**