# Введение

Тексты программ приведены частично. Полные тексты находятся в приложении. Нумерация строк в публикуемых фрагментах полностью совпадает с порядком следования строк в файле. Например, в следующем фрагменте кода из текста удалены две пустые строки (или комментарии).

```
16  }
19  {
```

Во всех текстах оставлены авторские комментарии.

Ссылки на строки исходного текста осуществляются с помощью знака «№» (номер).

Строки исходного текста программ печатаются следующим образом:

- номера строк в исходном тексте подсвечены,
- строки программы печатаются жирным шрифтом.

```
8       srt_arr[s_key] = s_val;
```

Если строка исходного текста слишком большая, то она переносится на следующую строку без дублирования номера строки, например:

```
 1  gawk --re-interval -f c:/awklib/pawk.mk
10_tch_kaf_tohand._aw > 10_tch_kaf_tohand.awk
```

Все элементы программы (переменные, функции, процедуры и т.д.) в основном тексте выделяются жирным шрифтом, например, функция **tbl_sort**, переменная **s_fld**.

# Используемые таблицы

Изучение структуры базы данных **Moodle** вызвало определенные трудности, вызванные тем, что ее описание практически отсутствует.

В описании (очень скромном) базы данных на сайте **moodle.org** говорится следующее:

«*База данных Moodle содержит около 200 таблиц, и может быть довольно сложной на первый взгляд. Хорошей новостью является то, что вам не обязательно понимать все сразу.*

*Например, есть восемь таблиц, которые называются forum_something. Если вам интересен модуль «форум», то, очевидно, вы должны разобраться в этих таблицах, изучить их поля и связи между ними. Но если вам форум не нужен, вы можете забыть о них.*

*Таким образом можно выделить около 50 основных таблиц. Но хорошая новость в том, что даже здесь они распадаются на группы, которые можно изучать независимо друг от друга*».

Каким образом мы должны «*разобраться в этих таблицах*» в описании не сказано.

После применения метода проб и ошибок, изучения базы данных с помощью **phpmyadmin** и других специфических действий, характерных для метода «научного тыка», нам удалось определить таблицы из которых мы можем получить необходимую информацию.

Для работы нам понадобятся следующие таблицы.

1. **mdl_user** – пользователи системы
2. **mdl_course** – курсы
3. **mdl_course_categories** – категории курсов (в нашем случае, кафедры, факультеты и т.д.
4. **mdl_user_lastaccess** – последний доступ пользователя к курсу.
5. **mdl_log** – логи системы, в которых фиксируется каждое действие пользователя.
6. **mdl_role_assignments** – возможные роли пользователя в различных элементах **Moodle**.
7. **mdl_context** – вспомогательная таблица, определяющая роль пользователя в конкретном курсе, модуле, блоке и т.д.

Подробная информация о таблицах и их связях. приведена в описании структуры таблиц и SQL- запросов, которые выбирают из таблиц всю необходимую информацию.

В описании полей каждой таблицы указана следующая информация:

1. Название поля.
2. Тип поля.
3. Может ли поле иметь значение **Null**.
4. Значение по умолчанию.

Элементы описания поля разделяются двоеточием. Используемые в запросах поля выделены жирным и прокомментированы.

# Таблица mdl_user

В таблице **mdl_user** определены следующие поля.
**id : int(10) : Нет :** – *идентификатор записи таблицы.*
auth : varchar(20) : Нет : db
confirmed : tinyint(1) : Нет : 0
policyagreed : tinyint(1) : Нет : 0
deleted : tinyint(1) : Нет : 0
mnethostid : bigint(10) : Нет : 0
username : varchar(100) : Нет :
password : varchar(32) : Нет :
idnumber : varchar(255) : Нет :
**firstname : varchar(100) : Нет :** – *имя пользователя*
**lastname : varchar(100) : Нет :** – *фамилия пользователя*
email : varchar(100) : Нет :
emailstop : tinyint(1) : Нет : 1
icq : varchar(15) : Нет :
skype : varchar(50) : Нет :
yahoo : varchar(50) : Нет :
aim : varchar(50) : Нет :
msn : varchar(50) : Нет :
phone1 : varchar(20) : Нет :
phone2 : varchar(20) : Нет :
institution : varchar(40) : Нет :
department : varchar(30) : Нет :
address : varchar(70) : Нет :
city : varchar(20) : Нет :
country : char(2) : Нет :
lang : varchar(30) : Нет : en_utf8
theme : varchar(50) : Нет :
timezone : varchar(100) : Нет : 99
firstaccess : int(10) : Нет : 0
lastaccess : int(10) : Нет : 0
lastlogin : int(10) : Нет : 0
currentlogin : int(10) : Нет : 0

lastip : varchar(15) : Нет :
secret : varchar(15) : Нет :
picture : tinyint(1) : Нет : 0
url : varchar(255) : Нет :
description : text : Да : NULL
mailformat : tinyint(1) : Нет : 1
maildigest : tinyint(1) : Нет : 0
maildisplay : tinyint(2) : Нет : 0
htmleditor : tinyint(1) : Нет : 1
ajax : tinyint(1) : Нет : 1
autosubscribe : tinyint(1) : Нет : 1
trackforums : tinyint(1) : Нет : 0
timemodified : int(10) : Нет : 0
trustbitmask : int(10) : Нет : 0
imagealt : varchar(255) : Да : NULL
screenreader : tinyint(1) : Нет : 0

## Таблица mdl_course

В таблице **mdl_course** определены следующие поля.
**id : int(10) : Нет :** – *идентификатор записи таблицы.*
**category : int(10) : Нет : 0** – *идентификатор категории курса (в нашем случае – идентификатор кафедры).*
sortorder : int(10) : Нет : 0
password : varchar(50): Нет :
**fullname : varchar(254): Нет :** – *полное название курса.*
**shortname : varchar(100): Нет :** – *краткое название курса.*
idnumber : varchar(100): Нет :
summary : text: Да : NULL
format : varchar(10): Нет : topics
showgrades : smallint(2) : Нет : 1
modinfo : longtext: Да : NULL
newsitems : smallint(5) : Нет : 1
teacher : varchar(100): Нет : Teacher
teachers : varchar(100): Нет : Teachers
student : varchar(100): Нет : Student
students : varchar(100): Нет : Students
guest : tinyint(2) : Нет : 0
startdate : int(10) : Нет : 0

enrolperiod : int(10) : Нет : 0
numsections : smallint(5) : Нет : 1
marker : int(10) : Нет : 0
maxbytes : int(10) : Нет : 0
showreports : int(4) : Нет : 0
**visible : int(1) : Нет : 1** – *виден ли курс студентам (1-да, 0-нет).*
hiddensections : int(2) : Нет : 0
groupmode : int(4) : Нет : 0
groupmodeforce : int(4) : Нет : 0
defaultgroupingid : bigint(10) : Нет : 0
lang : varchar(30): Нет :
theme : varchar(50): Нет :
cost : varchar(10): Нет :
currency : char(3): Нет : USD
timecreated : int(10) : Нет : 0
timemodified : int(10) : Нет : 0
metacourse : int(1) : Нет : 0
requested : int(1) : Нет : 0
restrictmodules : int(1) : Нет : 0
expirynotify : tinyint(1) : Нет : 0
expirythreshold : int(10) : Нет : 0
notifystudents : tinyint(1) : Нет : 0
enrollable : tinyint(1) : Нет : 1
enrolstartdate : int(10) : Нет : 0
enrolenddate : int(10) : Нет : 0
enrol : varchar(20): Нет :
defaultrole : int(10) : Нет : 0

## Таблица mdl_course_categories

В таблице **mdl_course_categories** определены следующие поля.
**id : int(10) : Нет :** – *идентификатор записи*.
**name : varchar(255): Нет :** – *название категории.*
description : text: Да : NULL
**parent : int(10) : Нет : 0** – *идентификатор родительской категории.*
sortorder : int(10) : Нет : 0

**coursecount : int(10) : Нет : 0** – *количество курсов в категории.*
visible : tinyint(1): Нет : 1
timemodified : int(10) : Нет : 0
**depth : int(10) : Нет : 0** – *длина пути, в поле path.*
**path : varchar(255): Нет :** – *полный путь от главной категории к текущей, номера категорий разделяются символом «/». Например, путь **«/3/4/10»** означает, что текущая категория (10) имеет родительскую категорию 4, категория 3 – родительская для категории 4.*
theme : varchar(50): Да : NULL

## Таблица mdl_user_lastaccess

В таблице **mdl_user_lastaccess** определены следующие поля.
**id : int(10) : Нет :** – *идентификатор записи.*
**userid : int(10) : Нет : 0** – *идентификатор пользователя.*
**courseid : int(10) : Нет : 0** – *идентификатор курса.*
**timeaccess : int(10) : Нет : 0** – *время последнего доступа пользоваделя к курсу.*

## Таблица mdl_log

В таблице **mdl_log** определены следующие поля.
**id : int(10) : Нет :** – *идентификатор записи.*
time : int(10) : Нет : 0
**userid : int(10) : Нет : 0** – *идентификатор пользователя.*
ip : varchar(15): Нет :
**course : int(10) : Нет : 0** – *идентификатор курса.*
module : varchar(20): Нет :
cmid : int(10) : Нет : 0
action : varchar(40): Нет :
url : varchar(100): Нет :
info : varchar(255): Нет :

## Таблица mdl_role_assignments

В таблице **mdl_role_assignments** определены следующие поля.
**id : int(10) : Нет :** – *идентификатор записи.*
**roleid : int(10) : Нет : 0** – *идентификатор роли пользователя.*
**contextid : int(10) : Нет : 0** – *тип контекста роли пользователя.*
**userid : int(10) : Нет : 0** – *идентификатор пользователя.*
hidden : int(1) : Нет : 0
timestart : int(10) : Нет : 0
timeend : int(10) : Нет : 0
timemodified : int(10) : Нет : 0
modifierid : int(10) : Нет : 0
enrol : varchar(20): Нет :
sortorder : int(10) : Нет : 0

## Таблица mdl_context

В таблице **mdl_context** определены следующие поля.
**id : int(10) : Нет :** – *идентификатор записи.*
**contextlevel : int(10) : Нет : 0:** – *тип контекста.*
**instanceid : int(10) : Нет : 0 :** – *поле связи с другими таблицами.*
path : varchar(255): Да : NULL :
depth : tinyint(2) : Нет : 0 :

# SQL запросы

Выбирать из базы необходимую информацию мы будем с помощью SQL-запросов.

Для выполнения запросов в нашей системе Moodle установлен специальный модуль – «пользовательские SQL-запросы». Адрес модуля:

http://docs.moodle.org/en/Custom_SQL_queries_report

Однако более перспективным кажется использование программы **phpmyadmin** или клиента (монитора) базы **mysql**.

Результаты работы SQL-запросов мы будем сохранять в текстовых файлах специального формата **csv**. Все поля в записях заключаются в кавычки и разделяются запятыми.

# Список курсов

Запрос на получение списка курсов достаточно прост и не требует подробных комментариев.

Оператор **SELECT** (№1-6) определяет выходную информацию, оператор **FROM** (№7-8) – используемые таблицы, а оператор **ORDER BY** (№9-10) – порядок сортировки.

```
1  SELECT
2  mdl_course.id AS 'ID',
3  mdl_course.visible AS 'VS',
4  mdl_course.category AS 'CT ID',
5  mdl_course.fullname AS 'FNM',
6  mdl_course.shortname AS 'SNM'
7  FROM
8  mdl_course
9  ORDER BY
10 mdl_course.fullname
```

В результате определены следующие поля (в процессе дальнейшей обработки информации мы будем обращаться к ним по этим именам).

- **ID** – идентификатор курса.
- **VS** – видимость курса.
- **CT ID** – категория курса.
- **FNM** – полное имя.
- **SNM** – короткое имя.

# Список категорий курсов

Список категорий курсов можно получить с помощью следующего запроса.

```
1  SELECT
2  mdl_course_categories.id AS 'ID',
3  mdl_course_categories.parent AS 'PARENT',
4  mdl_course_categories.name AS 'NAME',
5  mdl_course_categories.depth AS 'DEPTH',
6  mdl_course_categories.path AS 'PATH',
```

```
7   mdl_course_categories.coursecount 'NUM CR'
8   FROM
9   mdl_course_categories
10  ORDER BY
11  parent, id
```

В результате определены следующие поля.

- **ID** – идентификатор категории.
- **PARENT** – родительская категория.
- **NAME** – название категории
- **DEPTH** – длина пути, в поле path.
- **PATH** – полный путь от главной категории к текущей.
- **NUM CR** – количество курсов в категории.

## Количество пользователей курса

Определение количества пользователей курса – задача, которая с трудом поддается формализации, даже ее постановка сильно зависит от субъективных факторов. Мы решили пойти по простому пути – считаем пользователем курса любого, кто хоть раз на этот курс зашел. В этом случае, для определения числа пользователей можно использовать таблицу **mdl_user_ lastaccess.**

SQL-запрос имеет следующий вид.

```
1   SELECT
2   COUNT(mdl_user_lastaccess.id) AS 'NUM US',
3   mdl_user_lastaccess.courseid AS 'CR ID'
4   FROM
5   mdl_user_lastaccess
6   GROUP BY
7   mdl_user_lastaccess.courseid
```

В запросе два новых оператора SQL. Оператор **COUNT** (№2) подсчитывает количество величин (в данном случае, идентификаторов **ID**) со значением, не равным **NULL**, в строках, полученных при помощи команды **SELECT**.

Оператор **GROUP BY** (№2-3) определяет группировку при выборке данных. Все подсчеты выполняются для каждой группы отдельно. В данном случае, суммирование количества

записей (или значений идентификаторов **ID**) осуществляется для каждого курса отдельно.

Заметим, что если оператор **GROUP BY** не указан, то **COUNT** подсчитает общее количество записей в таблице.

В результате определены следующие поля.

- **NUM US** – количество пользователей.
- **CR ID** – идентификатор курса.

## Активность пользователей курса

Активность пользователей курса – это суммарное количество всех действий пользователей курса (естественно, в рамках курса).

SQL-запрос, подсчитывающий активность пользователей, практически не отличается от предыдущего запроса.

```
1  SELECT
2  COUNT(mdl_log.id) AS 'NUM ACT',
3  mdl_log.course AS 'CR ID'
4  FROM
5  mdl_log
6  WHERE
7  mdl_log.course > 1
8  GROUP BY
9  mdl_log.course
```

В результате определены следующие поля.

- **NUM ACT** – активность.
- **CR ID** – идентификатор курса.

## Определение преподавателей курса

Определение преподавателей всех курсов оказалось достаточно нетривиальной задачей. Пришлось использовать четыре таблицы (пятая таблица, **mdl_user_lastaccess**, потребовалась для определения последнего входа преподавателя на курс).

Оператор **SELECT** (№1-8) определяет поля результата запроса.

- **US ID** – идентификатор пользователя.
- **US LNM** – имя пользователя.
- **US FNM** – фамилия пользователя.
- **CR ID** – идентификатор курса.
- **LAST AC** – дата последнего входа пользователя на курс.
- **CR FNM** – полное название курса.
- **CT ID** – категория курса.

Отметим, что время измеряется в секундах от начала «эпохи UNIX» (1 января 1970 года) и представляет собой большое целое число. Функция **FROM_UNIXTIME** переводит это число в читаемый вид, например, **2011-03-29 14:42:52**.

```
1  SELECT
2  mdl_user.id AS 'US ID',
3  mdl_user.lastname AS 'US LNM',
4  mdl_user.firstname AS 'US FNM',
5  mdl_course.id AS 'CR_ID',
6  FROM_UNIXTIME(mdl_user_lastaccess.timeaccess) AS
'LAST AC',
7  mdl_course.fullname AS 'CR FNM',
8  mdl_course.category AS 'CT ID'
```

Оператор **FROM** (№9-14) определяет таблицы, из которых выбирается информация.

```
9  FROM
10 mdl_role_assignments,
11 mdl_context,
12 mdl_user,
13 mdl_course,
14 mdl_user_lastaccess
```

Оператор **WHERE** (№15-28) определяет условия выбора информации из нескольких таблиц. Используется также логический оператор **AND**, который показывает, что информация добавляется в результат запроса, только если все условия выполняются.

Мы выбираем только тех пользователей, которым назначена роль преподавателя (№18) в контексте курса (№16). Значение констант, определяющих контекст курса (50) и роль преподавателя (3) мы определили на основании предыдущего опыта и исследования исходных текстов программ.

В №20 связываются таблицы **mdl_role_assignments** (поле **contextid**) и **mdl_context** (поле **id**). Мы выбираем записи из таблицы **mdl_context**, которые соответствуют пользователям, соответствующим условиям в №16 и №18.

Таблица **mdl_context** нам нужна только для связи преподавателя с курсом, мы даже получили внятное объяснение (редкий случай) в описании базы данных.

*«Таблица **mdl_context** определяет контекст в Moodle, например, вся система, курс, конкретный элемент курса. Тип контекста задается переменной **contextlevel**, а переменная **instanceid** указывает таблицу, соответствующую типу контекста».*

Можно предположить, что если у нас тип контекста – курс, то переменная **instanceid** связывает таблицу **mdl_context** с таблицей **mdl_course**, а переменная связи в таблице **mdl_course** – идентификатор курса (**id**). В №22 эти таблицы связываются.

В №24 связываются таблицы **mdl_role_assignments** (поле **contextid**) и **mdl_user** (поле **id**). Из таблицы **mdl_user** мы возьмем имя и фамилию пользователя.

В №26 и №28 мы определяем записи в таблице **mdl_user_lastaccess**, в которых содержится информация о последнем входе каждого выбранного ранее преподавателя на каждый выбранный ранее курс. Для этого мы связываем таблицу **mdl_user_lastaccess** (поле **userid**) с таблицей **mdl_user** (поле **id**) и, через поле **courseid**, с таблицей **mdl_course** (поле **id**).

```
15  WHERE
16  (mdl_context.contextlevel=50)
17  AND
18  (mdl_role_assignments.roleid = 3)
19  AND
20  (mdl_role_assignments.contextid=mdl_context.id)
21  AND
22  (mdl_context.instanceid=mdl_course.id)
23  AND
24  (mdl_role_assignments.userid = mdl_user.id)
25  AND
```

```
26  (mdl_user_lastaccess.userid = mdl_user.id)
27  AND
28  (mdl_user_lastaccess.courseid = mdl_course.id)
```

Оператор **ORDER BY** (№29-32) определяет порядок сортировки выводимой информации. Сначала информация сортируется по идентификатору пользователя (№30), затем по последнему входу пользователя (№31), затем по идентификатору курса.

Сортировка по последнему входу пользователя осуществляется в порядке убывания, это задает ключевое слово **DESC**.

```
29  ORDER BY
30  mdl_user.id,
31  mdl_user_lastaccess.timeaccess DESC,
32  mdl_course.id
```

# Пользовательская библиотека функций awk

Мы получили информацию из базы данных Moodle и приступаем к ее обработке, используя, как всегда, универсальную программу работы с текстовыми файлами **awk**.

В процессе реализации задачи обработки информации выяснилось, что объем кода (строк программы) катастрофически растет. Кроме того, обработка информации достаточно унифицирована и отдельные блоки используются многократно.

Все это заставило нас реализовать идею, которая иногда возникала на протяжении нескольких лет – создать библиотеку функций, добавляемых в текст программы перед ее выполнением (пользовательскую библиотеку функций). Реализовать эту идею раньше мешала не природная лень (без которой программист легко превращается в педанта-буквоеда), а отсутствие задачи достаточной сложности и объема.

Итак, задача создания пользовательской библиотеки появилась. Реализация заняла значительно меньше времени, чем описание.

Поясним работу с пользовательской библиотекой функций на конкретном простом примере.

Пусть в рамках одной задачи (проекта) необходимо разработать две программы. Первая удаляет из каждой строчки входного файла лишние пробелы, а вторая еще и заменяет специальные символы **html** на последовательности, которые эти символы кодируют (например, кавычка кодируется последовательностью **"**).

Файл **ex1._aw** реализует первую задачу.

Строка, начинающаяся с символов **#libpath**, определяет путь к библиотеке функций (№1).

Строка, начинающаяся с символов **#include**, определяет имя файла, из которого мы вставляем информацию в исходную программу (№2).

```
1  #libpath c:/awklib
2  #include trim.f
4  {
5   print (trim($0));
6  }
```

Файл **ex2._aw** реализует вторую задачу.

```
1  #libpath c:/awklib
2  #include trim.f
3  #include HTML_spec.f
5  {
6    print HTML_spec(trim($0));
7  }
```

Функции, которые вставляются в исходные программы необходимо поместить в библиотеке.

Файл **trim.f** содержит функцию **trim,** которая удаляет из строки лишние пробелы (все пробелы в начале и в конце строки, любая последовательность пробелов в середине строки заменяется на один пробел).

```
1  function trim(str){
2    trim_str = str;
3    gsub(/^ +/,"",trim_str);
4    gsub(/ +$/,"",trim_str);
5    gsub(/ {2,}/," ",trim_str);
6    return trim_str;
```

```
7  }
```

Файл **HTML_spec.f** содержит функцию, которая кодирует специальные символы **html** (подробнее об этой функции см. [5]).

```
1  function HTML_spec(str,   s){
2    s = str;
3    gsub(/\&/,"\\\\\\&amp;",s);
4    gsub(/\"/,"\\\\\\&quot;",s);
5    gsub(/</,"\\\\\\&lt;",s);
6    gsub(/>/,"\\\\\\&gt;",s);
7    return s;
8  }
```

Файл ___**prj.bat** осуществляет «предпроцессорную» обработку файлов проекта, т.е. вставляет в тексты программ содержимое файлов из строк #**include**.

```
1  gawk --re-interval -f c:/awklib/pawk.mk ex1._aw >
ex1.awk
2  gawk --re-interval -f c:/awklib/pawk.mk ex2._aw >
ex2.awk
```

За добавление строчек из файлов отвечает программа (на языке **awk**) **pawk.mk.** Эта программа очень проста и в особых комментариях не нуждается.

```
1  function trim(str){
2    trim_str = str;
3    gsub(/^ +/,"",trim_str);
4    gsub(/ +$/,"",trim_str);
5    gsub(/ {2,}/," ",trim_str);
6    return trim_str;
7  }
9  function print_file(fn){
10   while(r = getline s < fn){
11     if(r == -1){
12       print "ERROR: " fn " - " ERRNO;
13       return -1;
14     }
15     print s
16   }
17   return 0;
18 }
```

```
20  BEGIN{
21    lp = "";
22  }
24  /^#libpath/{
25    s = $0;
26    gsub(/^#libpath/,"",s);
27    lp = trim(s) "/";
28    next;
29  }
31  /^#include/{
32    s = $0;
33    gsub(/^#include/,"",s);
34    fn = lp trim(s);
35    print_file(fn);
36    next;
37  }
39  {
40    print $0;
41  }
```

В результате получаем два готовых к выполнению программных файла.

Файл **ex1.awk**.

```
1   function trim(str){
2     trim_str = str;
3     gsub(/^ +/,"",trim_str);
4     gsub(/ +$/,"",trim_str);
5     gsub(/ {2,}/," ",trim_str);
6     return trim_str;
7   }
10  {
11   print (trim($0));
12  }
```

Файл **ex2.awk**.

```
1   function trim(str){
2     trim_str = str;
3     gsub(/^ +/,"",trim_str);
4     gsub(/ +$/,"",trim_str);
```

```
5     gsub(/ {2,}/," ",trim_str);
6     return trim_str;
7   }
9   function HTML_spec(str,   s){
10    s = str;
11    gsub(/\&/,"\\\\\\&amp;",s);
12    gsub(/\"/,"\\\\\\&quot;",s);
13    gsub(/</,"\\\\\\&lt;",s);
14    gsub(/>/,"\\\\\\&gt;",s);
15    return s;
16  }
19  {
20    print HTML_spec(trim($0));
21  }
```

Ниже мы опишем все функции, используемые в нашем проекте.

## Функции работы с файлами csv

Результаты работы SQL-запросов мы сохраняли в текстовых файлах специального формата **csv**. Все поля в записях заключаются в кавычки и разделяются запятыми. Фрагмент такого файла представлен ниже.

```
"NUM ACT","CR ID"
"23222","2"
"2199","11"
"26442","14"
```

Файлы такого типа хорошо понимают электронные таблицы. Для просмотра этих файлов мы использовали электронную таблицу **calc** пакета **Open Office**.

Мы определили три функции, обрабатывающие записи формата **csv**.

Функция **csv_get_rec** записывает все поля записи в массив **arr_res** и возвращает количество полей.

```
1   function csv_get_rec(str,arr_res,    n){
2     n = split(str,arr_res, "\",\"");
3     gsub(/^"/,"",arr_res[1]);
4     gsub(/"$/,"",arr_res[n]);
```

```
5     return n;
6   }
```

Функция **csv_make_rec** составляет запись (строку) из элементов массива **arr_fld** (**n_fld** – количество элементов) и возвращает эту строку.

```
1   function csv_make_rec(arr_fld,n_fld,   i,s){
2     s = "\"" arr_fld[1] "\"";
3     for(i=2; i<=n_fld; i++)
4       s = s "," "\"" arr_fld[i] "\"";
5     return s;
6   }
```

Функция **csv_newsep** переводит запись вида

```
"f1","f2", ... ,"fn"
```

в строку

```
f1 sep f2 sep … sep fn
```

Возвращается эта строка.

```
1   function csv_newsep(str,sep,    n,tmp_a,i,s){
2     n = csv_get_rec(str,tmp_a);
3     s = tmp_a[1]
4     for(i=2; i<=n; i++) s = s sep tmp_a[i];
5     return s;
6   }
```

# Функции работы с таблицами

Функции работы с таблицами реализуют основные возможности обработки информации, обычно применяемые в системах управления базами данных.

Таблицы читаются и записываются в файлы **csv**. Одно из полей определяется как ключевое и должно иметь уникальное значение для каждой записи.

Первая строчка таблицы содержит имена полей.

Для описания всех функций мы будем использовать тестовую таблицу (файл **csv**), текст файла приведен ниже.

```
"FN","DT","I"
"Иванов","2011-03-02","2"
"Иванов","2011-03-02","6"
"Иванов","2011-03-01","5"
```

```
"Иванов","2011-03-01","1"
"Петров","2011-03-02","4"
"Петров","2011-03-01","3"
```

Функция **tbl_new** создает новую таблицу из файла **csv**.

Массивы **a_ind, a_data** предназначены для хранения информации. Для каждой таблицы создаются свои массивы, вся информация из них удаляется (№2-3). При создании таблицы массив **a_data** остается пустым.

Параметр **s_fld** должен содержать список имен полей, разделенных знаком «#» (такой способ описания полей выбран для упрощения написания текстов программ, кавычки в строковые переменные вносить не очень удобно).

Параметр **s_key** – это имя ключевого поля.

Функция **tbl_new** записывает в массив **a_ind** следующую информацию

**a_ind["rec_total"]** – количество записей в таблице (при создании равно 0).

**a_ind["fld_list"]** – список имен полей.

**a_ind["fld_total"]** – количество полей в записи.

**a_ind["key_index"]** – номер ключевого поля в списке полей.

Кроме того, для каждого поля в массиве определяется запись вида:

**a_ind**["*имя поля*"] = *номер поля в списке*.

Функция возвращает **1**, если имя ключевого поля отсутствует в списке полей, в противном случае – **0**.

```
1  function tbl_new(a_ind, a_data, s_fld, s_key,
tmp_a,i,n,k){
2    delete a_ind;
3    delete a_data;
4    a_ind["rec_total"] = 0;
5    a_ind["fld_list"] = s_fld;
6    n = split(s_fld,tmp_a,"#");
7    if(n==0) return 1;
8    for(i=1; i<=n; i++)
9      a_ind[tmp_a[i]] = i;
10   a_ind["fld_total"] = n;
11   k = 0;
12   for(i=1; i<=n; i++)
```

```
13      if(s_key == tmp_a[i]) k = i;
14    if(k==0) return 1;
15    a_ind["key_index"] = k;
16    return 0;
17  }
```

Тестовую таблицу можно создать с помощью следующего вызова функции.

**tbl_new(indTST,dataTST,"FN#DT#I","I");**

При этом в массив **indTST** будет записана следующая информация.

**indTST["rec_total"]** = 0
**indTST["fld_total"]** = 3
**indTST["fld_list"]** = FN#DT#I
**indTST["key_index"]** = 3
**indTST["FN"]** = 1
**indTST["DT"]** = 2
**indTST["I"]** = 3

Функция **tbl_read** читает таблицу из файла. Первая строка файла должна содержать список имен полей.

**fn** – имя входного файла, **s_key** – имя ключевого поля.

Для создания таблицы вызывается функция **tbl_new**.

Функция возвращает **-1**, если произошел сбой при чтении файла, **1**, если имя ключевого поля отсутствует в списке полей, в противном случае – **0**.

```
1   function tbl_read(a_ind, a_data, s_key, fn,
s,tmp_a,i,n,r){
2     r = getline s < fn;
3     if(r == -1){
4         print "ERROR: " fn " - " ERRNO;
5         return -1;
6       }
7     s = csv_newsep(s ,"#");
8     if(tbl_new(a_ind, a_data, s, s_key)) return 1;
9     while(r = getline s < fn){
10      if(r == -1){
11        print "ERROR: " fn " - " ERRNO;
12        return -1;
```

```
13      }
14      n = csv_get_rec(s,tmp_a);
15      if(n < a_ind["key_index"]) return 1;
16      tbl_put_rec(a_ind, a_data,
tmp_a[a_ind["key_index"]], tmp_a);
17    }
18    close(fn);
19    return 0;
20  }
```

Тестовую таблицу можно прочитать из файла с помощью следующего вызова функции.

**tbl_read(indTST,dataTST,"I","ex.csv");**

При этом в массивы **indTST** и **dataTST** будет записана следующая информация.

```
indTST[1]=2    dataTST[2]="Иванов","2011-03-02","2"
indTST[2]=6    dataTST[6]="Иванов","2011-03-02","6"
indTST[3]=5    dataTST[5]="Иванов","2011-03-01","5"
indTST[4]=1    dataTST[1]="Иванов","2011-03-01","1"
indTST[5]=4    dataTST[4]="Петров","2011-03-02","4"
indTST[6]=3    dataTST[3]="Петров","2011-03-01","3"
```

Функция **tbl_get_rec** читает запись, соответствующую значению ключевого поля.

Параметр **s_key** – значение ключевого поля, значения всех полей записывается в массив **a_val**.

Функция возвращает **1**, если запись отсутствует в таблице, в этом случае в массив **a_val** записывается только значение ключевого поля.

Если запись в таблице найдена, то заполняются все элементы массива **a_val** и возвращается **0**.

```
1   function tbl_get_rec(a_ind,a_data,s_key,a_val,  i){
2     if(s_key in a_data){
3       csv_get_rec(a_data[s_key],a_val);
4       return 0;
5     }
6     delete a_val;
7     for(i=1;i<=a_ind["fld_total"];i++) a_val[i] = "";
8     a_val[a_ind["key_index"]] = s_key;
9     return 1;
```

```
10  }
```

Первую запись в таблице можно получить с помощью следующего вызова функции.

**tbl_get_rec(indTST,dataTST,"2",tmp_tst);**

В массив **tmp_tst** будет записана следующая информация.

```
tmp_tst[1]="Иванов"
tmp_tst[2]="2011-03-02"
tmp_tst[3]="2"
```

Функция **tbl_put_rec** заносит запись в таблицу.

Параметр **s_key** – значение ключевого поля, значения всех полей должны быть записаны в массиве **a_val**.

Функция возвращает **1**, если запись отсутствует в таблице, в этом случае в таблицу заносится новая запись.

Если запись в таблице найдена, то записывается вся информация из массива **a_val** (запись обновляется) и возвращается **0**.

```
1   function tbl_put_rec(a_ind, a_data, s_key, a_val,
s_data){
2     a_val[a_ind["key_index"]] = s_key;
3     s_data = csv_make_rec(a_val,a_ind["fld_total"]);
4     if(s_key in a_data){
5       a_data[s_key] = s_data;
6       return 0;
7     }
8     a_data[s_key] = s_data;
9     a_ind[++a_ind["rec_total"]] = s_key;
10    return 1;
11  }
```

Функция **tbl_sort_arr** сортирует массив **arr** (количество элементов – **nr**).

Если параметр **stp** равен **"DESC"**, то сортировка осуществляется по убыванию, в противном случае – по возрастанию.

Функция возвращает **1**, если количество элементов массива меньше двух, в противном случае – **0**.

```
1   function tbl_sort_arr(arr,nr,stp,     i,j,v){
```

```
2     if(nr<=1) return 1;
3     for(i=1; i<nr; i++)
4       for(j=i+1; j<=nr; j++)
5         if(((stp=="DESC") && (arr[i]<arr[j])) ||
6            ((stp!="DESC") && (arr[i]>arr[j])) ){
7           v = arr[i];
8           arr[i] = arr[j];
9           arr[j] = v;
10        }
11    return 0;
12  }
```

Функция **tbl_sort_fld** сортирует отдельное поле (имя поля – **s_fld**) в таблице, это поле должно состоять из подполей, разделенных строкой **sep**.

Если параметр **tp** равен **"DESC"**, то сортировка осуществляется по убыванию, в противном случае – по возрастанию.

```
1   function tbl_sort_fld(a_ind,a_data,s_fld,sep,tp,
tmp_a,n,s,i){
2     for(i=1; i<=a_ind["rec_total"]; i++){
3       tbl_get_rec(a_ind,a_data,a_ind[i],tmp_a);
4       s = tmp_a[a_ind[s_fld]];
5       n = split(s,arr,sep);
6       tbl_sort_arr(arr,n,tp);
7       s = arr[1];
8       for(j=2; j<=n; j++) s = s sep arr[j];
9       tmp_a[a_ind[s_fld]] = s;
10      tbl_put_rec(a_ind,a_data,a_ind[i],tmp_a);
11    }
12  }
```

Функция **tbl_sort**. сортирует таблицу по списку полей (параметр **s_fld**). Поля в списке разделяются символом "#".

**s_stp** – список типов сортировки для каждого поля (типы сортировки также разделяются символом "#").

```
1   function tbl_sort(a_ind,a_data,s_fld,s_stp,
2         pri,a_fld,a_stp,k,n,ai,aj,nr,i,j,si,sj,v){
```

```
3     n = split(s_fld,a_fld,"#");
4     split(s_stp,a_stp,"#");
5     nr = a_ind["rec_total"];
6     if(nr<=1) return 1;
7     for(i=1; i<nr; i++)
8       for(j=i+1; j<=nr; j++){
9         tbl_get_rec(a_ind,a_data,a_ind[i],ai);
10        tbl_get_rec(a_ind,a_data,a_ind[j],aj);
11        pri = 0;
12        for(k=1; k<=n; k++){
13          si = ai[a_ind[a_fld[k]]];
14          sj = aj[a_ind[a_fld[k]]];
15          if(si == sj) continue;
16          if(((a_stp[k]=="DESC") && (si<sj)) ||
17             ((a_stp[k]!="DESC") && (si>sj)) )
18            pri = 1;
19          break;
20        }
21        if(pri){
22          v = a_ind[i];
23          a_ind[i] = a_ind[j];
24          a_ind[j] = v;
25        }
26      }
27    return 0;
28  }
```

Вызов функции **tbl_sort(indTST,dataTST,"FN#DT#I", "ASC#DESC#0")** сортирует таблицу следующим образом

Сначала по полю **"FN"** (по возрастанию), затем по полю **"DT"** (по убыванию), затем по полю **"I"** (по возрастанию).

Функция **tbl_print** выводит таблицу в файл **fn**.

```
1   function tbl_print(a_ind, a_data,fn,    n,tmp_a,s){
2     n = split(a_ind["fld_list"],tmp_a,"#");
3     s = csv_make_rec(tmp_a,n);
4     if(fn != "") print s > fn;
5             else print s;
```

```
6     for(n=1; n<=a_ind["rec_total"]; n++)
7       if(fn != "") print a_data[a_ind[n]] > fn;
8               else print a_data[a_ind[n]];
9   }
```

Функция **tbl_debug_print** печатает содержимое таблицы в режиме отладки.

```
1   function tbl_debug_print(a_ind, a_data,    i){
2     for(i in a_ind)
3       if(i !~ /^[0-9]/)
4         if (fn != "")
printf("a_ind[%s]=%s\n",i,a_ind[i]) > fn
5                 else
printf("a_ind[%s]=%s\n",i,a_ind[i]);
6     if (fn != "") print "----------------------" > fn
7             else print "----------------------";
8     for(i=1; i<=a_ind["rec_total"]; i++)
9       if (fn != "") printf("[%s]=%s -> %s\n",
i,a_ind[i],a_data[a_ind[i]]) > fn
10              else printf("[%s]=%s -> %s\n",
i,a_ind[i],a_data[a_ind[i]]);
11  }
```

# Другие функции

Функция **HTML_spec** кодирует специальные символы **html** (см. выше).

Функция **read_ptn** читает шаблон отчета, в который выводятся результаты обработки информации (подробнее об этой функции см. [5]).

```
1   function read_ptn(fn,  i,r,s){
2     ptn_count = 0;
3     while(r = getline s < fn){
4       if(r == -1){
5         print "ERROR: " fn " - " ERRNO;
6         return -1;
7       }
```

```
8       if(s ~ /^<!##########/)
9         ptn_count++;
10      else
11        ptn_src[ptn_count] = ptn_src[ptn_count] s
"\n";
12    }
13    close(fn);
14    ptn_count++;
15    for(i=0; i<ptn_count; i++)
16      ptn_cur[i] = ptn_src[i];
17  }
```

Функция **read_strings** читает из файла массив строк, к этим строкам можно обращаться по индексу (имени строки).

```
1   function read_strings(fn,srt_arr,
s_key,s_val,r){
2     while(r = getline s_key < fn){
3       if(r == -1){
4         print "ERROR: " fn " - " ERRNO;
5         return -1;
6       }
7       r = getline s_val < fn
8       srt_arr[s_key] = s_val;
9     }
10    return 0;
11  }
```

# Обработка информации

В каталоге **work** приложения представлены все, файлы, которые мы использовали в процессе работы.

В подкаталоге **baza** находятся следующие результаты SQL-запросов.

- **cr_act_cnt.csv** – активность пользователей курса.
- **cr_cat.csv** – список категорий курсов.
- **cr_la_cnt.csv** – количество пользователей курса.
- **cr_tch_la.csv** – преподаватели курсов.

В подкаталоге **man_certif** находится таблица сертификатов (**sert.csv**) преподавателей, полученная вручную.

В подкаталоге **man_course** находятся следующие таблицы.

- **cr.csv** – результат SQL-запроса «список курсов». Таблица должна быть проверена и откорректирована вручную, мы находили в ней «левые записи».
- **cr_info_fin.csv** – которая определяет является ли курс активным. Решение принимает человек, результаты заносятся в специальное поле таблицы.

В подкаталоге **man_kaf** находится таблица **tch_kaf.csv**, в которую заносится информация о преподавателях всех кафедр, извлеченная из базы данных **Moodle**. При формировании таблицы требуется ручная обработка информации.

В подкаталоге **prj** находятся все файлы **_aw** (тексты программ, требующие предпроцессорной обработки перед выполнением), командный файл (**___prj.bat**), осуществляющий эту обработку и командные файлы вызова всех программ.

Текст файла **___prj.bat**.

```
1  gawk --re-interval -f c:/awklib/pawk.mk
10_tch_kaf_tohand._aw > 10_tch_kaf_tohand.awk
2  gawk --re-interval -f c:/awklib/pawk.mk
20_cr_tch._aw > 20_cr_tch.awk
3  gawk --re-interval -f c:/awklib/pawk.mk
30_cr_tohand._aw > 30_cr_tohand.awk
4  gawk --re-interval -f c:/awklib/pawk.mk
40_tch_info._aw > 40_tch_info.awk
5  gawk --re-interval -f c:/awklib/pawk.mk
50_cat_info._aw > 50_cat_info.awk
6  gawk --re-interval -f c:/awklib/pawk.mk out_tch._aw
> out_tch.awk
7  gawk --re-interval -f c:/awklib/pawk.mk
out_tch_kaf._aw > out_tch_kaf.awk
8  gawk --re-interval -f c:/awklib/pawk.mk
out_crs_kaf._aw > out_crs_kaf.awk
```

В подкаталоге **sql** находятся тексты всех SQL-запросов.

В подкаталоге **_site_ptn** находятся шаблоны отчетов.

Подкаталог **_html** предназначен для вывода информации в виде набора **html**-страниц.

В главном каталоге приложения приведены все программы и командные файлы, их вызывающие. Командные файлы нужно выполнять в определенной последовательности (они пронумерованы с шагом 10).

# Алгоритм работы

Перед началом обработки информации необходимо выполнить все SQL-запросы, сохранить их результаты и разместить их в соответствующих каталогах. Затем необходимо подготовить все таблицы, для формирования которых требуется ручная обработка информации.

**1.** Вызывается файл **10_tch_kaf_tohand.bat**, который определяет связь преподавателей и кафедр. Результат записывается в таблицу **man_kaf/tch_kaf_to_hand.csv**. После редактирования этого файла нужно сформировать таблицу **man_kaf/tch_kaf.csv**. Такое определение преподавателей – временная мера, в дальнейшем предполагается получать списки преподавателей из базы данных отдела кадров.

**2.** Файл **20_cr_tch.bat** осуществляет формирование списков курсов каждого преподавателя (таблица **tchcrla.csv**) и списков преподавателей каждого курса (**crtchla.csv**).

**3.** Файл **30_cr_tohand.bat** готовит таблицу (**man_course/ cr_info_hand.csv**), которая предназначена для определения вручную статуса курса (является ли курс активным). Результат должен быть представлен в таблице **man_course/ cr_info_fin.csv**.

**4.** Файл **40_tch_info.bat** формирует таблицу (**out_tch.csv**) с полной информацией о преподавателе, она включает количество курсов и активных курсов каждого преподавателя. Если количество активных курсов не равно нулю, то преподаватель считается «работающим».

**5.** Файл **50_cat_info.bat** осуществляет подсчет количества курсов и преподавателей для всех кафедр и категорий более высокого уровня, Активные курсы и работающие преподаватели выделяются отдельно. Выводится стартовая страница отчета – файл **0.html**.

**6.** В произвольном порядке вызываются файлы, формирующие выходную информацию:
**out_crs_kaf.bat** – списки курсов каждой кафедры.
**out_tch.bat** – списки курсов каждого преподавателя.
**out_tch_kaf.bat** – списки преподавателей каждой кафедры.

Перейдем к подробному описанию всех программных файлов.

# Определение преподавателей кафедр

Программный файл **10_tch_kaf_tohand._aw** определяет связь преподавателей и кафедр.

Эта программа вызывается с помощью командного файла **10_tch_kaf_tohand.bat**.

```
1  gawk --re-interval -f 10_tch_kaf_tohand.awk
baza\cr_tch_la.csv > report.txt
```

В №1-13 в текст программы включаются все необходимые функции.

```
1  #libpath c:/awklib
2  #include csv_get_rec.f
3  #include csv_make_rec.f
4  #include csv_newsep.f
5  #include tbl_sort_arr.f
6  #include tbl_new.f
7  #include tbl_put_rec.f
8  #include tbl_get_rec.f
9  #include tbl_read.f
10 #include tbl_print.f
11 #include tbl_sort_fld.f
12 #include tbl_sort.f
13 #include tbl_debug_print.f
```

В №17 создается таблица **man_kaf/tch_kaf_to_hand.csv** (в нее записываются результаты), содержащая следующие поля.

- **KEY** – ключевое поле, в него записывается идентификатор пользователя и идентификатор категории, разделенные знаком подчеркивания.
- **US_ID** – идентификатор пользователя.

- **US_NM** – Фамилия и имя пользователя.
- **US_CAT** – категория пользователя.
- **CAT_NM** – название категории.

В №20 из файла **man_course/cr.csv** читается таблица (список курсов), содержащая следующие поля.

- **ID** – идентификатор курса (ключевое поле).
- **VS** – видимость курса.
- **CT ID** – категория курса.
- **FNM** – полное имя.
- **SNM** – короткое имя.

В №23 из файла **baza/cr_cat.csv** читается таблица (категории курсов), содержащая следующие поля.

- **ID** – идентификатор категории.
- **PARENT** – родительская категория.
- **NAME** – название категории
- **DEPTH** – длина пути, в поле path.
- **PATH** – полный путь от главной категории к текущей.
- **NUM CR** – количество курсов в категории.

```
15  BEGIN{
17    tbl_new(indUS, dataUS,
"KEY#US_ID#US_NM#US_CAT#CAT_NM", "KEY");
20    tbl_read(indCR,dataCR,"ID","man_course/cr.csv");
23    tbl_read(indCAT,dataCAT,"ID","baza/cr_cat.csv");
24  }
```

В №26-56 обрабатываются записи входного файла (**baza/cr_tch_la.csv**), который содержит следующие поля.

- **US ID** – идентификатор пользователя.
- **US LNM** – фамилия пользователя.
- **US FNM** – имя пользователя
- **CR ID** – идентификатор курса.
- **LAST AC** – последний вход преподавателя на курс.
- **CR FNM** – полное название курса.
- **CT ID** – категория курса.

Запись, содержащая имена полей не обрабатывается (№26-29).

```
26  NR == 1{
27    next;
```

```
28  }
```

В №34-38 строка входного файла разбивается на отдельные поля.

Из таблицы **man_course/cr.csv** выбирается запись, соответствующая идентификатору курса **cr_id** (№42). Из этой записи определяется идентификатор категории курса и записывается в переменную **cat_id** (№43). Затем (№44) из таблицы **baza/cr_cat.csv** читается запись, соответствующая значению ключевого поля **cat_id**, из этой записи выбирается название категории (№45).

```
33  {
34    csv_get_rec($0,arr_line)
35    us_id = arr_line[1];
36    cr_id = arr_line[4];
37    us_nm = arr_line[2] " " arr_line[3];
38    cr_nm = arr_line[6];
42    r = tbl_get_rec(indCR,dataCR,cr_id,tmp_a);
43    cat_id = tmp_a[indCR["CT ID"]];
44    tbl_get_rec(indCAT,dataCAT,cat_id,tmp_a);
45    cat_nm = tmp_a[indCAT["NAME"]];
```

В №47 формируется значение ключевого поля для выходной таблицы.

В №50-57 в выходную таблицу заносится очередная запись.

```
47    key = us_id "_" cat_id;
50    tbl_get_rec(indUS,dataUS,key,tmp_a);
51    tmp_a[indUS["US_ID"]] = us_id;
52    tmp_a[indUS["US_NM"]] = us_nm;
53    tmp_a[indUS["US_CAT"]] = cat_id;
54    tmp_a[indUS["CAT_NM"]] = cat_nm;
55    tbl_put_rec(indUS,dataUS,key,tmp_a);
56  }
```

Выходная таблица сортируется по пользователю и категории (№59) и записывается в файл (№60).

```
58  END{
59    tbl_sort(indUS,dataUS,"US_ID#US_CAT","ASC#ASC");
60    tbl_print(indUS,dataUS,
"man_kaf/tch_kaf_to_hand.csv");
```

```
63  }
```

# Определение курсов преподавателя и преподавателей курса

Программный файл **20_cr_tch._aw** формирует список преподавателей и их курсов, а также список курсов и их преподавателей.

Эта программа вызывается с помощью командного файла **20_cr_tch.bat**.

```
1   gawk --re-interval -f 20_cr_tch.awk
baza\cr_tch_la.csv> report.txt
```

В №1-13 в текст программы включаются все необходимые функции (см. описание программы **10_tch_kaf_tohand._aw**).

В №17 создается таблица **tchcrla.csv**, содержащая следующие поля.

**US_ID** – идентификатор пользователя (ключевое поле).
**US_NM** – фамилия и имя пользователя.
**US_CR** – курсы пользователя.

Все курсы преподавателя записываются в поле **US_CR**, информация о каждом курсе отделяется символом "#". Для каждого курса записывается последний вход преподавателя и идентификатор курса (через пробел), например

"**2011-06-11 409#2011-06-10 195**".

В №18 создается таблица **crtchla.csv**, содержащая следующие поля.

**CR_ID** – идентификатор курса (ключевое поле).
**CR_NM** – название курса.
**CR_US** – преподаватели курса (структура этого поля аналогична **US_CR**).

```
15  BEGIN{
16    tbl_new(indUS, dataUS, "US_ID#US_NM#US_CR",
"US_ID");
17    tbl_new(indCR, dataCR, "CR_ID#CR_NM#CR_US",
"CR_ID");
18  }
```

В №20-48 обрабатываются записи входного файла (**baza/cr_tch_la.csv**), который содержит следующие поля.

- **US ID** – идентификатор пользователя.
- **US LNM** – фамилия пользователя.
- **US FNM** – имя пользователя
- **CR ID** – идентификатор курса.
- **LAST AC** – последний вход преподавателя на курс.
- **CR FNM** – полное название курса.
- **CT ID** – категория курса.

Запись, содержащая имена полей не обрабатывается (№20-22).

```
20  NR == 1{
21    next;
22  }
```

В №28-33 строка входного файла разбивается на отдельные поля.

В таблице **tchcrla.csv** читается запись, соответствующая текущему идентификатору пользователя (№36), в поле «курсы преподавателя» дописывается идентификатор текущего курса и последний вход преподавателя на этот курс (№38-39), затем запись обновляется (№40).

```
27  {
28    csv_get_rec($0,arr_line);
29    us_id = arr_line[1];
30    cr_id = arr_line[4];
31    us_nm = arr_line[2] " " arr_line[3];
32    cr_nm = arr_line[6];
33    last_ac = substr(arr_line[5],1,10);
36    r = tbl_get_rec(indUS,dataUS,us_id,tmp_a);
37    tmp_a[indUS["US_NM"]] = us_nm;
38    if(r == 0) tmp_a[indUS["US_CR"]] =
tmp_a[indUS["US_CR"]] "#";
39    tmp_a[indUS["US_CR"]] = tmp_a[indUS["US_CR"]]
last_ac " " cr_id;
40    tbl_put_rec(indUS,dataUS,us_id,tmp_a);
```

В таблице **crtchla.csv** читается запись, соответствующая текущему идентификатору курса (№43), в поле «преподаватели курса» дописывается идентификатор текущего преподавателя и

его последний вход на этот курс (№45-46), затем запись обновляется (№47).

```
43    r = tbl_get_rec(indCR,dataCR,cr_id,tmp_a);
44    tmp_a[indCR["CR_NM"]] = cr_nm;
45    if(r == 0) tmp_a[indCR["CR_US"]] =
tmp_a[indCR["CR_US"]] "#";
46    tmp_a[indCR["CR_US"]] = tmp_a[indCR["CR_US"]]
last_ac " " us_id;
47    tbl_put_rec(indCR,dataCR,cr_id,tmp_a);
48  }
49
```

В таблице **tchcrla.csv** поле «курсы преподавателя» сортируется в порядке убывания (т.е. по последнему входу преподавателя, №51). Затем вся таблица сортируется в порядке убывания по этому полю (№52).

```
50  END{
51    tbl_sort_fld(indUS,dataUS,"US_CR","#","DESC");
52    tbl_sort(indUS,dataUS,"US_CR","DESC");
```

В таблице **crtchla.csv** поле «преподаватели» сортируется в порядке убывания (т.е. по последнему входу преподавателя, №53). Затем вся таблица сортируется в порядке убывания по этому полю (№54).

```
53    tbl_sort_fld(indCR,dataCR,"CR_US","#","DESC");
54    tbl_sort(indCR,dataCR,"CR_US","DESC");
```

В №56-57 таблицы записываются в выходные файлы.

```
56    tbl_print(indUS,dataUS,"tchcrla.csv");
57    tbl_print(indCR,dataCR,"crtchla.csv");
60  }
```

# Информация о курсах

Программный файл **30_cr_tohand._aw** собирает всю информацию о каждом курсе и формирует таблицу, на основании которого человек принимает решение об активности курса.

Эта программа вызывается с помощью командного файла **30_cr_tohand.bat**.

```
1   gawk --re-interval -f 30_cr_tohand.awk > report.txt
```

В №1-13 в текст программы включаются все необходимые функции (см. описание программ **10_tch_kaf_tohand._aw** и **20_cr_tch._aw**).

В №15 дополнительно включается функция чтения массива строк. Строки читаются из файла ___**str.**___ в №18.

В №20 создается выходная таблица **man_course/ cr_info_hand.csv**, содержащая следующие поля.

- **CR_TP** – тип курса (активный или нет).
- **CR_ID** – идентификатор курса (ключевое поле).
- **CR_LA** – последний вход преподавателя на курс.
- **CR_US** – количество пользователей курса.
- **CR_ACT** – активность на курсе.
- **CR_NM** – название курса.
- **CR_AD** – адрес курса в системе **Moodle** (**URL**).

```
15  #include read_strings.f
17  BEGIN{
18    read_strings("___str.___",GB_STR);
20    tbl_new(indCR,dataCR,
"CR_TP#CR_ID#CR_LA#CR_US#CR_ACT#CR_NM#CR_AD","CR_ID");
```

Из файла **crtchla.csv** читается таблица (№23), в выходную таблицу переписывается информация о последнем входе преподавателя на курс (№24-32).

```
23    tbl_read(indSRV,dataSRV,"CR_ID","crtchla.csv");
24    for(i=1; i<=indSRV["rec_total"]; i++){
25      s_key = indSRV[i];
26      tbl_get_rec(indSRV,dataSRV,s_key,tmp_a);
27      s = tmp_a[indSRV["CR_US"]];
28      s = substr(s,1,10);
29      tbl_get_rec(indCR,dataCR,s_key,tmp_a);
30      tmp_a[indCR["CR_LA"]] = s;
31      tbl_put_rec(indCR,dataCR,s_key,tmp_a);
32    }
```

Из файла **man_course/cr.csv** читается таблица (№35), в выходную таблицу переписывается информация о названии курса. В поле **CR_TP** записывается информация о видимости курса (№36-45).

```
35    tbl_read(indSRV,dataSRV,"ID",
"man_course/cr.csv");
```

```
36    for(i=1; i<=indSRV["rec_total"]; i++){
37      s_key = indSRV[i];
38      tbl_get_rec(indSRV,dataSRV,s_key,tmp_a);
39      s = tmp_a[indSRV["FNM"]];
40      s1 = tmp_a[indSRV["VS"]];
41      tbl_get_rec(indCR,dataCR,s_key,tmp_a);
42      tmp_a[indCR["CR_NM"]] = s;
43      tmp_a[indCR["CR_TP"]] = s1;
44      tbl_put_rec(indCR,dataCR,s_key,tmp_a);
45    }
```

Из файла **baza/cr_la_cnt.csv** читается таблица (№48), в выходную таблицу переписывается информация о количестве пользователей курса (№49-56).

```
48    tbl_read(indSRV, dataSRV, "CR ID",
"baza/cr_la_cnt.csv");
49    for(i=1; i<=indSRV["rec_total"]; i++){
50      s_key = indSRV[i];
51      tbl_get_rec(indSRV,dataSRV,s_key,tmp_a);
52      s = tmp_a[indSRV["NUM US"]];
53      tbl_get_rec(indCR,dataCR,s_key,tmp_a);
54      tmp_a[indCR["CR_US"]] = s;
55      tbl_put_rec(indCR,dataCR,s_key,tmp_a);
56    }
```

Из файла **baza/cr_act_cnt.csv** читается таблица (№59), в выходную таблицу переписывается информация об активности на курсе (№60-67).

```
59    tbl_read(indSRV,dataSRV,"CR
ID","baza/cr_act_cnt.csv");
60    for(i=1; i<=indSRV["rec_total"]; i++){
61      s_key = indSRV[i];
62      tbl_get_rec(indSRV,dataSRV,s_key,tmp_a);
63      s = tmp_a[indSRV["NUM ACT"]];
64      tbl_get_rec(indCR,dataCR,s_key,tmp_a);
65      tmp_a[indCR["CR_ACT"]] = s;
66      tbl_put_rec(indCR,dataCR,s_key,tmp_a);
67    }
```

В №70-77 определяются и записываются в выходную таблицу адреса курсов в системе **Moodle**.

В №79 выходная таблица выводится в файл.

```
70    for(i=1; i<=indCR["rec_total"]; i++){
71      s_key = indCR[i];
72      tbl_get_rec(indCR,dataCR,s_key,tmp_a);
74      s = GB_STR["COURSE_URL"] s_key;
75      tmp_a[indCR["CR_AD"]] = s;
76      tbl_put_rec(indCR,dataCR,s_key,tmp_a);
77    }
79    tbl_print(indCR,dataCR,
"man_course/cr_info_hand.csv");
82  }
```

# Информация о преподавателе

Программный файл **40_tch_info._aw** собирает всю информацию о преподавателе и формирует таблицу для отчетов.

Эта программа вызывается с помощью командного файла **40_tch_info.bat**

```
1  gawk --re-interval -f 40_tch_info.awk > report.txt
```

В №1-13 в текст программы включаются все необходимые функции (см. описание программ **10_tch_kaf_tohand._aw**, **20_cr_tch._aw** и **30_cr_tohand._aw)**.

В №21 создается выходная таблица **out_tch.csv**, содержащая следующие поля.

**US_ID** – идентификатор пользователя (ключевое поле).
**CR_ALL** – количество курсов преподавателя.
**CR_G** – количество активных курсов преподавателя.
**US_SRT** – дата выдачи сертификата.
**US_CAT** – категория (кафедра) преподавателя.
**US_NM** – фамилия имя преподавателя.
**US_CR** – список курсов преподавателя.

```
16  BEGIN{
17    tbl_new(indUS,dataUS,"US_ID#CR_ALL#CR_G#US_SRT#
US_CAT#US_NM#US_CR", "US_ID");
```

Из файла **tchcrla.csv** читается таблица (№21), в выходную таблицу переписывается фамилия, имя преподавателя и его список курсов (№22-31).

```
21    tbl_read(indSRV,dataSRV,"US_ID","tchcrla.csv");
22    for(i=1; i<=indSRV["rec_total"]; i++){
23      s_key = indSRV[i];
24      tbl_get_rec(indSRV,dataSRV,s_key,tmp_a);
25      s1 = tmp_a[indSRV["US_NM"]];
26      s2 = tmp_a[indSRV["US_CR"]];
27      tbl_get_rec(indUS,dataUS,s_key,tmp_a);
28      tmp_a[indUS["US_NM"]] = s1;
29      tmp_a[indUS["US_CR"]] = s2;
30      tbl_put_rec(indUS,dataUS,s_key,tmp_a);
31    }
```

Из файла **man_certif/cert.csv** читается таблица (№35) в выходную таблицу добавляется дата выдачи сертификата (№36-47). Если преподаватель, получивший сертификат, не определен во входных таблицах, из таблицы сертификатов выбираются имя и фамилия.

```
35    tbl_read(indSRV,dataSRV,"USID",
"man_certif/cert.csv");
36    for(i=1; i<=indSRV["rec_total"]; i++){
37      srt_us_id = indSRV[i];
38      tbl_get_rec(indSRV,dataSRV,srt_us_id,tmp_srt);
39      s_cd = tmp_srt[indSRV["CERDT"]];
40      s_cd = substr(s_cd,1,10);
41      s_nm = tmp_srt[indSRV["TNM"]];
42      tbl_get_rec(indUS,dataUS,srt_us_id,tmp_us);
43      tmp_us[indUS["US_SRT"]] = s_cd;
44      if(tmp_us[indUS["US_NM"]] == "")
45        tmp_us[indUS["US_NM"]] = s_nm;
46      tbl_put_rec(indUS,dataUS,srt_us_id,tmp_us);
47    }
```

Из файла **man_kaf/tch_kaf.csv** читается таблица (№52), в выходную таблицу добавляется категория (кафедра) преподавателя (№53-63). Если преподаватель не определен во входных таблицах, из таблицы категорий выбираются имя и фамилия.

```
52    tbl_read(indSRV,dataSRV,"US_ID",
man_kaf/tch_kaf.csv");
53    for(i=1; i<=indSRV["rec_total"]; i++){
54      us_id = indSRV[i];
55      tbl_get_rec(indSRV,dataSRV,us_id,tmp_kaf);
56      us_cat = tmp_kaf[indSRV["US_CAT"]];
57      s_nm = tmp_srt[indSRV["US_NM"]];
58      tbl_get_rec(indUS,dataUS,us_id,tmp_us);
59      tmp_us[indUS["US_CAT"]] = us_cat;
60      if(tmp_us[indUS["US_NM"]] == "")
61        tmp_us[indUS["US_NM"]] = s_nm;
62      tbl_put_rec(indUS,dataUS,us_id,tmp_us);
63    }
```

Из файла **man_course/cr_info_fin.csv** читается таблица (№69), в выходную таблицу добавляются общее количество курсов и количество активных курсов преподавателя (№70-88).

Информация о том, активный курс или нет, записана в поле **CR_TP**.

```
69    tbl_read(indSRV,dataSRV,"CR_ID",
"man_course/cr_info_fin.csv");
70    for(j=1; j<=indUS["rec_total"]; j++){
71      us_id = indUS[j];
72      tbl_get_rec(indUS,dataUS,us_id,tmp_us);
73      s_us_cr = tmp_us[indUS["US_CR"]];
74      n = split(s_us_cr,a_us_cr,"#");
75      cr_all = 0;
76      cr_g = 0;
77      for(k=1; k<=n; k++){
78        cr_all++;
79        split(a_us_cr[k],a1);
80        cr_id = a1[2];
81        tbl_get_rec(indSRV,dataSRV,cr_id,tmp_cr);
82        cr_tp = tmp_cr[indSRV["CR_TP"]];
83        if(cr_tp != 0) cr_g++;
84      }
85      tmp_us[indUS["CR_ALL"]] = cr_all;
86      tmp_us[indUS["CR_G"]] = cr_g;
```

```
87      tbl_put_rec(indUS,dataUS,us_id,tmp_us);
88    }
```

Выходная таблица сортируется по кафедре, последнему входу преподавателя (по убыванию) и фамилии (№90). Затем выходная таблица выводится в файл (№91).

```
90    tbl_sort(indUS,dataUS,"US_CAT#US_CR#US_NM",
"ASC#DESC#ASC");
91    tbl_print(indUS,dataUS,"out_tch.csv");
94  }
```

# Категории курсов

Программный файл **50_cat_info._aw** осуществляет подсчет количества курсов и преподавателей для всех кафедр и категорий более высокого уровня, Активные курсы и работающие преподаватели выделяются отдельно. Выводится стартовая страница отчета – файл **0.html**.

С формальной точки зрения не совсем понятно, какую категорию можно считать кафедрой. Информация, полученная из таблиц базы данных, не позволяет ответить на этот вопрос однозначно. Мы будем считать, что категория является кафедрой, *если к ней можно привязать курсы*. В противном случае, категория кафедрой не является, даже если в ее названии есть слово «кафедра».

Эта программа вызывается с помощью командного файла **50_cat_info.bat**.

```
1   gawk --re-interval -f 50_cat_info.awk > report.txt
```

Программа использует технологию формирования отчетов, подробное описание этой технологии представлено в [5].

Шаблон для формирования выходного файла (**0.__h**) приведен ниже.

```
1   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
2   <html>
3   <head>
4   <meta http-equiv="Content-Type" content="text/html;
```

```
charset=utf-8">
 5   <title>Преподаватели и курсы ЦДО ХНАГХ</title>
 6   </head>
 7   <body>
 8   <h2 align="center">Преподаватели и курсы ЦДО
ХНАГХ</h2>
 9   <table width="100%" border="1" cellspacing="2"
cellpadding="2">
10     <tr>
11       <td rowspan="2" align="center" valign="top">
<strong>№</strong></td>
12       <td rowspan="2" align="center" valign="top" >
<strong>Название</strong></td>
13       <td colspan="2" align="center" valign="top" >
<strong>Преподаватели</strong></td>
14       <td colspan="2" align="center" valign="top" >
<strong>Курсы</strong></td>
15     </tr>
16     <tr>
17       <td align="center" valign="top" >
<strong>Всего</strong></td>
18       <td align="center" valign="top"
><strong>Работающие</strong></td>
19       <td align="center" valign="top" >
<strong>Всего</strong></td>
20       <td align="center" valign="top" >
<strong>Активные</strong></td>
21     </tr>
22   <!########### end of part 0>
23     <tr>
24       <td align="center" valign="top" >==NN==</td>
25       <td align="left" valign="top" >==CAT_NM==</td>
26       <td align="right" valign="top" > ==TCH_ALL==
</td>
27       <td align="right" valign="top" >==TCH_G==</td>
28       <td align="right" valign="top" >==CR_ALL==</td>
29       <td align="right" valign="top" >==CR_G==</td>
30     </tr>
```

```
31  <!########## end of part 1>
32  </table>
33  </body>
34  </html>
```

В №1-13 в текст программы включаются все необходимые функции (см. описание программ **10_tch_kaf_tohand._aw**, **20_cr_tch._aw**, **30_cr_tohand._aw** и **40_tch_info._aw)**.

В №15-17 дополнительно включаются:

- функция кодировки специальных символов **html**,
- функция чтения шаблона отчета,
- функция чтения массива строк.

```
15  #include HTML_spec.f
16  #include read_ptn.f
17  #include read_strings.f
```

В №26 создается временная выходная таблица, содержащая следующие поля.

**ID** – идентификатор категории (ключевое поле).
**PARENT** – родительская категория.
**NAME** – название категории.
**PATH** – полный путь от главной категории к текущей, номера категорий разделяются символом «/». Например, путь **«/3/4/10»** означает, что текущая категория (10) имеет родительскую категорию 4, категория 3 – родительская для категории 4.
**DEPTH** – длина пути, в поле **PATH**.
**ISKAF** – определяет, является ли категория кафедрой (**1**) или категорией более высокого уровня (**0**).
**TCH_ALL** – всего преподавателей в категории.
**TCH_G** – количество работающих преподавателей.
**CR_ALL** – всего курсов в категории.
**CR_G** – количество активных курсов.

```
19  BEGIN{
20    gbl_tch_all = 0;
21    gbl_tch_g   = 0;
22    gbl_cr_all  = 0;
23    gbl_cr_g    = 0;
26    tbl_new(indKAF,dataKAF,"ID#PARENT#NAME#PATH#
DEPTH#ISKAF#TCH_ALL#TCH_G#CR_ALL#CR_G","ID");
```

Из файла **baza/cr_cat.csv** читается таблица (№29), в выходную таблицу переписывается следующая информация (№30-44):

- родительская категория,
- название категории,
- полный путь от главной категории к текущей,
- длина пути.

Остальные поля таблицы заполняются нулями.

```
29    tbl_read(indSRV,dataSRV,"ID","baza/cr_cat.csv");
30    for(i=1; i<=indSRV["rec_total"]; i++){
31      cat_id = indSRV[i];
32      tbl_get_rec(indKAF,dataKAF,cat_id,tmp_kaf);
33      tbl_get_rec(indSRV,dataSRV,cat_id,tmp_a);
34      tmp_kaf[indKAF["PARENT"]]  =
tmp_a[indSRV["PARENT"]];
35      tmp_kaf[indKAF["NAME"]]    =
tmp_a[indSRV["NAME"]];
36      tmp_kaf[indKAF["PATH"]]    =
tmp_a[indSRV["PATH"]];
37      tmp_kaf[indKAF["DEPTH"]]    =
tmp_a[indSRV["DEPTH"]];
38      tmp_kaf[indKAF["ISKAF"]]   = 0;
39      tmp_kaf[indKAF["TCH_ALL"]] = 0;
40      tmp_kaf[indKAF["TCH_G"]]   = 0;
41      tmp_kaf[indKAF["CR_ALL"]]  = 0;
42      tmp_kaf[indKAF["CR_G"]]    = 0;
43      tbl_put_rec(indKAF,dataKAF,cat_id,tmp_kaf);
44    }
```

Из файла **out_tch.csv** читается таблица (№47).

Для текущей категории определяется число общее число преподавателей и количество работающих преподавателей (№50-57).

Общее число преподавателей и количество работающих преподавателей подсчитываются для ВУЗа в целом, затем эти величины определяются для всех родительских категорий (№61-65).

```
47    tbl_read(indSRV,dataSRV,"US_ID","out_tch.csv");
49    for(i=1; i<=indSRV["rec_total"]; i++){
```

```
50        gbl_tch_all++;
51        us_id = indSRV[i];
52        tbl_get_rec(indSRV,dataSRV,us_id,tmp_a);
53        cat_id = tmp_a[indSRV["US_CAT"]];
54        cr_g   = tmp_a[indSRV["CR_G"]];
55        tch_g = 0;
56        if(cr_g > 0) tch_g = 1;
57        gbl_tch_g += tch_g;
58        tbl_get_rec(indKAF,dataKAF,cat_id,tmp_kaf);
59        np = split(tmp_kaf[indKAF["PATH"]],a_path,"/");
60        for(j=2; j <=np; j++){
61          tbl_get_rec(indKAF,dataKAF,a_path[j],
tmp_kaf);
62          tmp_kaf[indKAF["TCH_G"]] += tch_g;
63          tmp_kaf[indKAF["TCH_ALL"]]++;
64          tbl_put_rec(indKAF,dataKAF,a_path[j],
tmp_kaf);
65        }
66      }
```

В №69 и №72 таблицы читаются из файлов **man_course/cr.csv** и **man_course/cr_info_fin.csv**.

Для текущей категории определяется число общее курсов и количество активных курсов (№75-86).

Общее курсов и количество активных курсов подсчитываются для ВУЗа в целом, затем эти величины определяются для всех родительских категорий (№88-93).

```
69      tbl_read(indCR,dataCR,"ID","man_course/cr.csv");
72      tbl_read(indSRV,dataSRV,"CR_ID",
"man_course/cr_info_fin.csv");
74      for(i=1; i<=indCR["rec_total"]; i++){
75        gbl_cr_all++;
76        cr_id = indCR[i];
77        tbl_get_rec(indCR,dataCR,cr_id,tmp_cr);
78        cat_id = tmp_cr[indCR["CT ID"]];
79        tbl_get_rec(indSRV,dataSRV,cr_id,tmp_a);
80        cr_g = 0;
81        cr_tp = tmp_a[indSRV["CR_TP"]];
82        if(cr_tp > 0) cr_g = 1;
```

```
83        gbl_cr_g += cr_g;
84        tbl_get_rec(indKAF,dataKAF,cat_id,tmp_kaf);
85        tmp_kaf[indKAF["ISKAF"]] = 1;
86        tbl_put_rec(indKAF,dataKAF,cat_id,tmp_kaf);
87        np = split(tmp_kaf[indKAF["PATH"]],a_path,"/");
88        for(j=2; j <=np; j++){
89          tbl_get_rec(indKAF,dataKAF,a_path[j],
tmp_kaf);
90          tmp_kaf[indKAF["CR_G"]] += cr_g;
91          tmp_kaf[indKAF["CR_ALL"]]++;
92          tbl_put_rec(indKAF,dataKAF,a_path[j],
tmp_kaf);
93        }
94      }
```

В №96-163 происходит вывод стартовой страницы отчета.

Выходная таблица сортируется по полю **PATH** (№96).

В №100 осуществляется чтение списка строк из файла. Определены следующие строки

**COURSE_URL** – постоянная часть адреса курса:
*http://cdo.kname.edu.ua/course/view.php?id=*
**USER_URL** – постоянная часть адреса пользователя:
*http://cdo.kname.edu.ua/user/view.php?id=*
**MSG_ALL_AK** – служебное сообщение:
*Всего по академии*

Затем читается шаблон отчета **_site_ptn/0.__h** (№101) и выводится часть отчета **0**, которая не содержит переменной информации.

```
96      tbl_sort(indKAF,dataKAF,"PATH","ASC");
98      npp = 1;
99      fn = "_html/0.html";
100     read_strings("___str.___",GB_STR);
101     read_ptn("_site_ptn/0.__h");
102     printf(ptn_cur[0]) > fn;
```

В №104-116 выводится информация по ВУЗу в целом.

Заполняются следующие поля части **1** шаблона (№104-115):

"**==NN==**" – номер категории по порядку,

"**==CAT_NM==**" – название категории,
"**==TCH_ALL==**" – всего преподавателей,
"**==TCH_G==**" – количество работающих преподавателей,
"**==CR_ALL==**" – всего курсов,
"**==CR_G==**" – количество активных курсов.

В №116 выводится часть **1** шаблона.

```
104   ptn_cur[1] = ptn_src[1];
105   cat_nm  = "<b>" GB_STR["MSG_ALL_AK"] "</b>";
106   tch_all = gbl_tch_all;
107   tch_g   = gbl_tch_g;
108   cr_all  = gbl_cr_all;
109   cr_g    = gbl_cr_g;
110   gsub(/==NN==/,npp,ptn_cur[1]);
111   gsub(/==CAT_NM==/,cat_nm,ptn_cur[1]);
112   gsub(/==TCH_ALL==/,tch_all,ptn_cur[1]);
113   gsub(/==TCH_G==/,tch_g,ptn_cur[1]);
114   gsub(/==CR_ALL==/,cr_all,ptn_cur[1]);
115   gsub(/==CR_G==/,cr_g,ptn_cur[1]);
116   printf(ptn_cur[1]) > fn;
```

В №104-116 выводится информация для каждой категории.

Если категория не является кафедрой, то информация выводится жирным шрифтом и ссылки на другие **html**-страницы не определяются (№ 130-136).

В противном случае (№137-145) информация выводится обычным шрифтом, формируются ссылки на список курсов кафедры и список преподавателей кафедры.

```
119   for(i=1; i<=indKAF["rec_total"]; i++){
120     cat_id = indKAF[i];
121     if(cat_id == 0)continue;
122     tbl_get_rec(indKAF,dataKAF,cat_id,tmp_kaf);
123     tmp_kaf[indKAF["PARENT"]];
124     _cat_nm  = tmp_kaf[indKAF["NAME"]];
125     iskaf    = tmp_kaf[indKAF["ISKAF"]];
126     _tch_all = tmp_kaf[indKAF["TCH_ALL"]];
127     _tch_g   = tmp_kaf[indKAF["TCH_G"]];
128     _cr_all  = tmp_kaf[indKAF["CR_ALL"]];
129     _cr_g    = tmp_kaf[indKAF["CR_G"]];
```

```
130       if(iskaf == 0){
131         cat_nm  = "<b>" _cat_nm "</b>";
132         tch_all = "<b>" _tch_all "</b>";
133         tch_g   = "<b>" _tch_g "</b>";
134         cr_all  = "<b>" _cr_all "</b>";
135         cr_g    = "<b>" _cr_g "</b>";
136       }
137       else{
138         cat_nm = sprintf("<a
href=\"kc%4.4d.html\">%s</a>",cat_id,_cat_nm);
139         if(_tch_all != 0)
140           tch_all = sprintf("<a
href=\"kt%4.4d.html\">%s</a>", cat_id,_tch_all)
141         else tch_all = _tch_all;
142         tch_g   = _tch_g;
143         cr_all  = sprintf("<a
href=\"kc%4.4d.html\">%s</a>",cat_id,_cr_all);
144         cr_g    =  _cr_g;
145       }
```

Заполняются поля части **1** шаблона (№146-154) и эта часть шаблона выводится (№155).

```
146       dp = tmp_kaf[indKAF["DEPTH"]]
147       for(j=1; j<dp; j++) cat_nm = " ___ " cat_nm;
148       ptn_cur[1] = ptn_src[1];
149       gsub(/==NN==/,npp++,ptn_cur[1]);
150       gsub(/==CAT_NM==/,cat_nm,ptn_cur[1]);
151       gsub(/==TCH_ALL==/,tch_all,ptn_cur[1]);
152       gsub(/==TCH_G==/,tch_g,ptn_cur[1]);
153       gsub(/==CR_ALL==/,cr_all,ptn_cur[1]);
154       gsub(/==CR_G==/,cr_g,ptn_cur[1]);
155       printf(ptn_cur[1]) > fn;
156     }
158      printf(ptn_cur[2]) > fn;
163   }
```

# Вывод информации о преподавателе

Программный файл **out_tch._aw** осуществляет вывод всех курсов каждого преподавателя. Неактивные курсы выделяются серым цветом фона. Имя страницы отчета **tXXXXX.html** (**XXXXX** – идентификатор пользователя).

Эта программа вызывается с помощью командного файла **out_tch.bat**.

```
1  gawk --re-interval -f out_tch.awk > report.txt
```

Шаблон для формирования выходного файла (**tch_1.__h**) приведен ниже.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
5  <title>==US_NM==</title>
6  </head>
7  <body>
8  <a href="0.html">Стартовая страница</a> --->
9  <a href="==KAF_AD==">==KAF_NM==</a>
10 <h1 align="center"><a href="==US_AD=="
target="_blank">==US_NM==</a></h1>
11 <p align="center"><strong>Дата выдачи сертификата:
==CRT_DT==</strong></p>
12 <table width="100%" border="1" cellspacing="2"
cellpadding="2">
13   <tr>
14     <th scope="col">№</th>
15     <th scope="col">Название</th>
16     <th scope="col">Посл.<br>вход.<br>преп.<br>
</th>
17     <th scope="col">К-во<br>студ.</th>
18     <th scope="col">Актив-<br>ность</th>
19   </tr>
20 <!########### end of part 0>
21   <tr>
22     <td align="center" valign="top" bgcolor=
```

```
"==COL==">==NN==</td>
23      <td align="left" valign="top" bgcolor=
"==COL==><a href="==CR_AD==" target="_blank">
==CR_NM==</a></td>
24      <td align="center" valign="top" bgcolor=
"==COL==">==TCH_LA==</td>
25      <td align="right" valign="top" bgcolor=
"==COL==">==ST_NUM==</td>
26      <td align="right" valign="top" bgcolor=
"==COL==">==ACTIV==</td>
27    </tr>
28  <!########## end of part 1>
29  </table>
30  </body>
31  </html>
```

В №1-17 в текст программы включаются все необходимые функции (см. описание программы **50_cat_info._aw**).

В №20 из файла **_site_ptn/tch_1.__h** читается шаблон отчета.

В №26 из файла **out_tch.csv** читается таблица, содержащая следующие поля.

- **US_ID** – идентификатор пользователя (ключевое поле).
- **CR_ALL** – количество курсов преподавателя.
- **CR_G** – количество активных курсов преподавателя.
- **US_SRT** – дата выдачи сертификата.
- **US_CAT** – категория (кафедра) преподавателя.
- **US_NM** – фамилия и имя преподавателя.
- **US_CR** – список курсов преподавателя.

В №29 из файла **man_course/cr_info_fin.csv** читается таблица, содержащая следующие поля.

- **CR_TP** – тип курса (активный или нет).
- **CR_ID** – идентификатор курса (ключевое поле).
- **CR_LA** – последний вход преподавателя на курс.
- **CR_US** – количество пользователей курса.
- **CR_ACT** – активность на курсе.
- **CR_NM** – название курса.
- **CR_AD** – адрес курса в системе **Moodle** (**URL**).

В №32 из файла **man_kaf/tch_kaf.csv** читается таблица, содержащая следующие поля.

- **KEY** – идентификатор пользователя и идентификатор категории, разделенные знаком подчеркивания.
- **US_ID** – идентификатор пользователя (ключевое поле).
- **US_NM** – Фамилия и имя пользователя.
- **US_CAT** – категория пользователя.
- **CAT_NM** – название категории.

```
19  BEGIN{
20    read_strings("___str.___",GB_STR);
23    read_ptn("_site_ptn/tch_1.__h");
26    tbl_read(indUS,dataUS,"US_ID","out_tch.csv");
29    tbl_read(indCR,dataCR,"CR_ID",
"man_course/cr_info_fin.csv");
32    tbl_read(indKAF,dataKAF,"US_ID",
"man_kaf/tch_kaf.csv");
```

В №44 формируется имя выходного файла.

```
42    for(i=1; i<=indUS["rec_total"]; i++){
43      us_id = indUS[i];
44      fn = sprintf("_html/t%5.5d.html",us_id);
```

В №46-49 из таблицы **out_tch.csv** читаются список курсов преподавателя, фамилия и имя преподавателя, дата выдачи сертификата.

```
46      tbl_get_rec(indUS,dataUS,us_id,tmp_us);
47      us_cr  = tmp_us[indUS["US_CR"]];
48      us_nm  = tmp_us[indUS["US_NM"]];
49      us_srt  = tmp_us[indUS["US_SRT"]];
```

В №51-53 из таблицы **man_kaf/tch_kaf.csv** читаются категория пользователя и название категории.

В №54 формируется адрес списка преподавателей кафедры.

```
51      tbl_get_rec(indKAF,dataKAF,us_id,tmp_kaf);
52      cat_us = tmp_kaf[indKAF["US_CAT"]];
53      cat_nm = tmp_kaf[indKAF["CAT_NM"]];
54      cat_ad = sprintf("kt%4.4d.html",cat_us);
```

В №56-62 выводится информация о преподавателе.

Заполняются следующие поля части **0** шаблона:

"**==KAF_AD==**" – адрес списка преподавателей кафедры,

"**==KAF_NM==**" – название кафедры,
"**==US_NM==**" – фамилия и имя пользователя,
"**==US_AD==**" – адрес пользователя в системе **Moodle**,
"**==CRT_DT==**" – дата выдачи сертификата.

Затем часть **0** шаблона выводится (№62).

```
56      ptn_cur[0] = ptn_src[0];
57      gsub(/==KAF_AD==/,cat_ad,ptn_cur[0]);
58      gsub(/==KAF_NM==/,cat_nm,ptn_cur[0]);
59      gsub(/==US_NM==/,us_nm,ptn_cur[0]);
60      gsub(/==US_AD==/,GB_STR["USER_URL"] us_id,
ptn_cur[0]);
61      gsub(/==CRT_DT==/,us_srt,ptn_cur[0]);
62      printf(ptn_cur[0]) > fn;
```

Определяется список курсов преподавателя (№64), для каждого курса выделяется его идентификатор (№67) и последний вход преподавателя (№68).

В №69-73 для каждого курса преподавателя из таблицы **man_course/cr_info_fin.csv** читаются активность на курсе, количество пользователей курса, название курса, тип курса (активный или нет).

В №74-75 определяется цвет фона выводимой строки. Если курс активный, то цвет фона – белый (**#FFFFFF**), в противном случае – серый (**#CCCCCC**).

```
64      n = split(us_cr,a_cr,"#");
65      for(j=1; j<=n; j++){
66        split(a_cr[j],a_cr_la);
67        cr_id = a_cr_la[2];
68        la = a_cr_la[1];
69        tbl_get_rec(indCR,dataCR,cr_id,tmp_cr);
70        cr_act = tmp_cr[indCR["CR_ACT"]];
71        cr_us  = tmp_cr[indCR["CR_US"]];
72        cr_nm  = tmp_cr[indCR["CR_NM"]];
73        cr_tp  = tmp_cr[indCR["CR_TP"]];
74        col = "#FFFFFF";
75        if(cr_tp == 0) col = "#CCCCCC";
```

В №77-85 выводится информация о курсе.

Заполняются следующие поля части 1 шаблона:
"**==NN==**" – номер курса в списке,

"**==CR_NM==**" – название курса,
"**==CR_AD==**" – адрес курса в системе **Moodle**,
"**==TCH_LA==**" – последний вход преподавателя,
"**==ST_NUM==**" – количество студентов,
"**==ACTIV==**" – активность на курсе.
**"==COL=="** – цвет фона.

Затем часть **1** шаблона выводится (№88).

```
77          ptn_cur[1] = ptn_src[1];
78          gsub(/==NN==/,j,ptn_cur[1]);
79          gsub(/==CR_NM==/,cr_nm,ptn_cur[1]);
80          gsub(/==CR_AD==/,GB_STR["COURSE_URL"] cr_id,
ptn_cur[1]);
81          gsub(/==TCH_LA==/,la,ptn_cur[1]);
82          gsub(/==ST_NUM==/,cr_us,ptn_cur[1]);
83          gsub(/==ACTIV==/,cr_act,ptn_cur[1]);
84          gsub(/==COL==/,col,ptn_cur[1]);
85          printf(ptn_cur[1]) > fn;
87        }
88        printf(ptn_cur[2]) > fn;
89      }
90  }
```

# Вывод списка преподавателей кафедры

Программный файл **out_tch_kaf._aw** осуществляет вывод списка преподавателей кафедры. Неработающие преподаватели выделяются серым цветом фона. Имя страницы отчета **ktXXXX.html** (**XXXX** – идентификатор кафедры).

Эта программа вызывается с помощью командного файла **out_tch_kaf.bat**.

```
1   gawk --re-interval -f out_tch_kaf.awk > report.txt
```

Шаблон для формирования выходного файла (**tch_kaf.__h**) приведен ниже.

```
1   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
2   <html>
3   <head>
```

```
 4  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
 5  <title>==KAF_NM==</title>
 6  </head>
 7  <body>
 8  <a href="0.html">Стартовая страница</a>
 9  <h2 align="center">Список преподавателей</h2>
10  <h1 align="center">==KAF_NM==</h1>
11  <table width="100%" border="1" cellspacing="2"
cellpadding="2">
12    <tr>
13      <th scope="col">№</th>
14      <th scope="col">ФИО<br>преподавателя</th>
15      <th scope="col">Дата<br>сертиф.</th>
16      <th scope="col">Посл.<br>вход.<br>преп.<br>
</th>
17      <th scope="col">К-во<br>курсов</th>
18      <th scope="col">Из них<br>актив.</th>
19    </tr>
20  <!########## end of part 0>
21    <tr>
22      <td align="center" valign="top" bgcolor=
"==COL==">==NN==</td>
23      <td align="left" valign="top" bgcolor=
"==COL==><a href="==TCH_AD==">==TCH_NM==</a></td>
24      <td align="center" valign="top" bgcolor=
"==COL==">==TCH_CER==</td>
25      <td align="center" valign="top" bgcolor=
"==COL==">==TCH_LA==</td>
26      <td align="right" valign="top" bgcolor=
"==COL==">==CR_NUM==</td>
27      <td align="right" valign="top" bgcolor=
"==COL==">==CR_ACT==</td>
28    </tr>
29  <!########## end of part 1>
30  </table>
31  </body>
32  </html>
```

В №1-17 в текст программы включаются все необходимые функции (см. описание программ **50_cat_info._aw** и **out_tch._aw**).

В №23 из файла **_site_ptn/tch_kaf.__h** читается шаблон отчета.

В №26 из файла **out_tch.csv** читается таблица, содержащая следующие поля.

- **US_ID** – идентификатор пользователя (ключевое поле).
- **CR_ALL** – количество курсов преподавателя.
- **CR_G** – количество активных курсов преподавателя.
- **US_SRT** – дата выдачи сертификата.
- **US_CAT** – категория (кафедра) преподавателя.
- **US_NM** – фамилия и имя преподавателя.
- **US_CR** – список курсов преподавателя.

Таблица сортируется по категории, фамилии преподавателя и его последнему входу (№27).

В №30 из файла **man_kaf/tch_kaf.csv** читается таблица, содержащая следующие поля.

- **KEY** – идентификатор пользователя и идентификатор категории, разделенные знаком подчеркивания.
- **US_ID** – идентификатор пользователя (ключевое поле).
- **US_NM** – Фамилия и имя пользователя.
- **US_CAT** – категория пользователя.
- **CAT_NM** – название категории.

```
19  BEGIN{
20    read_strings("___str.___",GB_STR);
23    read_ptn("_site_ptn/tch_kaf.__h");
26    tbl_read(indUS,dataUS,"US_ID","out_tch.csv");
27    tbl_sort(indUS,dataUS,"US_CAT#US_NM#US_CR",
"0#0#DESC");
30    tbl_read(indKAF,dataKAF,"US_ID",
"man_kaf/tch_kaf.csv");
```

В №41-54 выводится заголовок отчета для каждой категории. Заполняется поле "**==KAF_NM==**" (название кафедры) части **0** шаблона, затем эта часть шаблона выводится.

```
38    fn = "";
39    nu = 0;
```

```
40    for(i=1; i<=indUS["rec_total"]; i++){
41      us_id = indUS[i];
42      tbl_get_rec(indUS,dataUS,us_id,tmp_us);
43      us_cat = tmp_us[indUS["US_CAT"]];
44      fn1 = sprintf("_html/kt%4.4d.html",us_cat);
45      if(fn != fn1){
46        if(fn != "") printf(ptn_cur[2]) > fn;
47        fn = fn1;
48        tbl_get_rec(indKAF,dataKAF,us_id,tmp_kaf);
49        kaf_nm  = tmp_kaf[indKAF["CAT_NM"]];
50        ptn_cur[0] = ptn_src[0];
51        gsub(/==KAF_NM==/,kaf_nm,ptn_cur[0]);
52        printf(ptn_cur[0]) > fn;
53        nu = 0;
54      }
```

В №56-62 из таблицы **out_tch.csv** выбирается следующая информация: фамилия и имя преподавателя, дата выдачи сертификата, общее количество курсов, количество работающих курсов.

Если у преподавателя нет активных курсов, то он считается «неработающим», информация о нем выделяется серым цветом фона (№63-64).

```
56        tch_nm   = tmp_us[indUS["US_NM"]];
57        tch_ad   = sprintf("t%5.5d.html",us_id);
58        tch_srt  = tmp_us[indUS["US_SRT"]];
59        tch_la   = substr(tmp_us[indUS["US_CR"]],
1,10);
60        cr_num   = tmp_us[indUS["CR_ALL"]];
61        cr_act   = tmp_us[indUS["CR_G"]];
63        col = "#FFFFFF";
64        if(cr_act == 0) col = "#CCCCCC";
65
```

В №66-75 выводится информация о преподавателе.

Заполняются следующие поля части **1** шаблона:

"**==NN==**" – номер по порядку,
"**==TCH_NM==**" – фамилия и имя преподавателя,
"**==TCH_AD==**" – адрес преподавателя в системе **Moodle**,
"**==TCH_CER==**" – дата выдачи сертификата,

"**==TCH_LA==**" – последний вход преподавателя в систему,
"**==CR_NUM==**" – количество курсов,
"**==CR_ACT==**" – количество активных курсов,
"**==COL==**" – цвет фона.

Затем часть **1** шаблона выводится.

```
66          ptn_cur[1] = ptn_src[1];
67          gsub(/==NN==/,++nu,ptn_cur[1]);
68          gsub(/==TCH_NM==/,tch_nm,ptn_cur[1]);
69          gsub(/==TCH_AD==/,tch_ad,ptn_cur[1]);
70          gsub(/==TCH_CER==/,tch_srt,ptn_cur[1]);
71          gsub(/==TCH_LA==/,tch_la,ptn_cur[1]);
72          gsub(/==CR_NUM==/,cr_num,ptn_cur[1]);
73          gsub(/==CR_ACT==/,cr_act,ptn_cur[1]);
74          gsub(/==COL==/,col,ptn_cur[1]);
75          printf(ptn_cur[1]) > fn;
77      }
78      printf(ptn_cur[2]) > fn;
80  }
```

# Вывод списка курсов кафедры

Программный файл **out_crs_kaf._aw** осуществляет вывод списка курсов кафедры. Неактивные курсы выделяются серым цветом фона. Имя страницы отчета **kcXXXX.html** (**XXXX** – идентификатор кафедры).

Эта программа вызывается с помощью командного файла **out_crs_kaf.bat**.

```
1   gawk --re-interval -f out_crs_kaf.awk > report.txt
```

Шаблон для формирования выходного файла (**crs_kaf.__h**) приведен ниже.

```
1   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
2   <html>
3   <head>
4   <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
5   <title>==KAF_NM==</title>
```

```
 6  </head>
 7  <body>
 8  <a href="0.html">Стартовая страница</a>
 9  <h2 align="center">Список курсов</h2>
10  <h1 align="center">==KAF_NM==</h1>
11  <table width="100%" border="1" cellspacing="2"
cellpadding="2">
12    <tr>
13      <th scope="col">№</th>
14      <th scope="col">Название</th>
15      <th
scope="col">Посл.<br>вход.<br>преп.<br></th>
16      <th scope="col">К-во<br>студ.</th>
17      <th scope="col">Актив-<br>ность</th>
18    </tr>
19  <!########## end of part 0>
20    <tr>
21      <td align="center" valign="top" bgcolor=
"==COL==">==NN==</td>
22      <td align="left" valign="top" bgcolor=
"==COL==><a href="==CR_AD==" target="_blank">
==CR_NM==</a></td>
23      <td align="center" valign="top" bgcolor=
"==COL==">==TCH_LA==</td>
24      <td align="right" valign="top" bgcolor=
"==COL==">==ST_NUM==</td>
25      <td align="right" valign="top" bgcolor=
"==COL==">==ACTIV==</td>
26    </tr>
27  <!########## end of part 1>
28  </table>
29  </body>
30  </html>
```

В №1-17 в текст программы включаются все необходимые функции (см. описание программ **50_cat_info._aw**, **out_tch._aw** и **out_tch_kaf._aw**).

В №23 из файла **_site_ptn/crs_kaf.__h** читается шаблон отчета.

В №26 из файла **man_course/cr.csv** читается таблица, содержащая следующие поля.

- **ID** – идентификатор курса (ключевое поле).
- **VS** – видимость курса.
- **CT ID** – категория курса.
- **FNM** – полное имя.
- **SNM** – короткое имя.

Таблица сортируется по категории и фамилии пользователя.

В №30 из файла **man_course/cr_info_fin.csv** читается таблица, содержащая следующие поля.

- **CR_TP** – тип курса (активный или нет).
- **CR_ID** – идентификатор курса (ключевое поле).
- **CR_LA** – последний вход преподавателя на курс.
- **CR_US** – количество пользователей курса.
- **CR_ACT** – активность на курсе.
- **CR_NM** – название курса.
- **CR_AD** – адрес курса в системе **Moodle** (**URL**).

В №33 из файла **baza/cr_cat.csv** читается таблица (категории курсов), содержащая следующие поля.

- **ID** – идентификатор категории.
- **PARENT** – родительская категория.
- **NAME** – название категории
- **DEPTH** – длина пути, в поле path.
- **PATH** – полный путь от главной категории к текущей.
- **NUM CR** – количество курсов в категории.

```
19  BEGIN{
20    read_strings("___str.___",GB_STR);
23    read_ptn("_site_ptn/crs_kaf.__h");
26    tbl_read(indCR,dataCR,"ID","man_course/cr.csv");
27    tbl_sort(indCR,dataCR,"CT ID#FNM","ASC#ASC");
30    tbl_read(indINF,dataINF,"CR_ID",
"man_course/cr_info_fin.csv");
33    tbl_read(indCAT,dataCAT,"ID","baza/cr_cat.csv");
```

В №43-57 выводится заголовок отчета для каждой категории. Заполняется поле "**==KAF_NM==**" (название кафедры) части **0** шаблона, затем эта часть шаблона выводится.

```
40    fn = "";
41    npp = 0;
42    for(i=1; i<=indCR["rec_total"]; i++){
43      cr_id  = indCR[i];
44      tbl_get_rec(indCR,dataCR,cr_id,tmp_cr);
45      cr_cat = tmp_cr[indCR["CT ID"]];
47      fn1 = sprintf("_html/kc%4.4d.html",cr_cat);
48      if(fn != fn1){
49        if(fn != "") printf(ptn_cur[2]) > fn;
50        fn = fn1;
51        tbl_get_rec(indCAT,dataCAT,cr_cat,tmp_kaf);
52        kaf_nm  = tmp_kaf[indCAT["NAME"]];
53        ptn_cur[0] = ptn_src[0];
54        gsub(/==KAF_NM==/,kaf_nm,ptn_cur[0]);
55        printf(ptn_cur[0]) > fn;
56        npp = 0;
57      }
```

В №58-64 из таблицы **man_course/cr_info_fin.csv** выбирается следующая информация: название курса, последний вход преподавателя на курс, количество пользователей курса, активность на курсе, тип курса (активный или нет), адрес курса в системе **Moodle**.

Если курс не является активным, информация о нем выделяется серым цветом фона (№66-67).

```
58      tbl_get_rec(indINF,dataINF,cr_id,tmp_inf);
59      cr_nm  = tmp_cr[indCR["FNM"]];
60      tch_la = tmp_inf[indINF["CR_LA"]];
61      st_num = tmp_inf[indINF["CR_US"]];
62      activ  = tmp_inf[indINF["CR_ACT"]];
63      cr_tp  = tmp_inf[indINF["CR_TP"]];
64      cr_ad = GB_STR["COURSE_URL"] cr_id;
66      col = "#FFFFFF";
67      if(cr_tp == 0) col = "#CCCCCC";
68
```

В №69-78 выводится информация о курсе.

Заполняются следующие поля части **1** шаблона:

"**==NN==**" – номер по порядку,

"**==CR_AD==**" – адрес курса в системе **Moodle**,

"**==CR_NM==**" – название курса,
"**==TCH_LA==**" – последний вход преподавателя в на курс,
"**==ST_NUM==**" – количество пользователей,
"**==ACTIV==**" – активность на курсе,
"**==COL==**" – цвет фона.

Затем часть **1** шаблона выводится.

```
69      ptn_cur[1] = ptn_src[1];
70      gsub(/==NN==/,++npp,ptn_cur[1]);
71      gsub(/==CR_AD==/,cr_ad,ptn_cur[1]);
72      gsub(/==CR_NM==/,cr_nm,ptn_cur[1]);
73      gsub(/==TCH_LA==/,tch_la,ptn_cur[1]);
74      gsub(/==ST_NUM==/,st_num,ptn_cur[1]);
75      gsub(/==ACTIV==/,activ,ptn_cur[1]);
76      gsub(/==COL==/,col,ptn_cur[1]);
77      printf(ptn_cur[1]) > fn;
78    }
80    printf(ptn_cur[2]) > fn;
82  }
```

# Заключение

В работе описан алгоритм получения статистической информации о курсах и преподавателях в **Moodle**. Подробно описаны все необходимые запросы к базе данных, программы обработки информации и шаблоны отчетов.

Следует отметить, что в работе представлена первая версия программного продукта, которая будет наверняка изменяться. Поэтому основное внимание мы уделили логике работы всех программ, изучив которую каждый сможет развивать систему в нужном ему направлении.

Описываемая в работе задача без сомнения принадлежит к классу задач принятия решений. Ее полная формализация и построение необходимых математических моделей – предмет будущих исследований.