грифами, что позволяет рассчитать их процентное содержание в фонде; выделены издания, коэффициент обеспеченности которых не удовлетворяет нормативу, и вычислено необходимое количество экземпляров этих изданий для ретроспективного комплектования; распределена литература для книговыдачи по семестрам и специальностям; отобрана литература по видам изданий для определения состава фонда.

Данная экспериментальная проверка показала, что создание электронной картотеки книгообеспеченности с помощью имеющихся в вузовских библиотеках программных средств в принципе возможно. При этом корректное заполнение полей электронного каталога позволит в будущем перейти на практически любую программу автоматизации библиотек без потери существующих данных.

Бочаров Б.П., Воеводина М.Ю

ГЛОБАЛЬНАЯ КОРРЕКТИРОВКА БАЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ПРОГРАММЫ AWK

Эту статью авторы адресуют коллегам, которые уже имеют некоторый опыт в ведении электронных каталогов (или других БД), уже испытали от этого изрядную долю восторгов и разочарований и хотели бы сделать решительный шаг для поправки своих дел. Особенно хочется подчеркнуть, что предлагаемая технология позволит ничего не потерять из того, что уже сделано.

Почему это для нас важно?

Итак, вы создали свою базу данных и с воодушевлением начинаете ее вести. Работа кропотливая, часто отдает рутиной, а выполнять ее надо быстро. Во-первых, потому, что очень хочется увидеть пользу от ведения БД. А это возможно только при достаточно большом объеме, так как никого (а в особенности вашего начальника) не впечатлит «быстрый поиск» в базе, насчитывающей всего лишь сотню записей.

Во-вторых, почувствовать преимущества ведения именно электронного каталога вы можете только в том случае, когда в нем отражена значительная часть вышего фонда (а в идеале - весь). В такой ситуации ошибки при заполнении самых совершенных входных форм неизбежны! И не надо считать, что они есть зло: все, что делается людьми должно быть усовершенствовано ими же.

Прибавьте к этому, что вы и ваши ближайшие коллеги осваивают новую область, что одна и та же база данных ведется несколькими сотрудниками (а вузы часто еще и студентов привлекают в такой работе в период летней практики), что записи могут создаваться в несколько этапов и пр. Ошибки появляются самые разнообразные и неожиданные: незаполненные поля, применение недопустимых символов, неправильно введенный текст, некорректная разбивка по подполям и т.д.

Как быть?

Можно годами просматривать на экране уже созданные записи и так и не «выловить» все ошибки, поскольку наш глаз видит далеко не все, что мы ему поручаем. Хотя этим «прогрессивным» методом поиска ошибок в основном-то все и пользуются. Авторы предлагают принципиально иной подход к этому вопросу и называют его глобальной корректировкой базы данных. Такой подход, конечно, требует привлечения к работе программистов.

Под глобальной корректировкой БД будем подразумевать корректировку содержимого полей, выполнение которой не связано с редактированием в формах самой БД.

И еще приятные неожиданности!

Кроме корректировки записей вы попутно можете решить некоторые другие свои насущные проблемы, казалось бы, далекие от проверки правильности заполнения полей БД. Например, вы можете создать свое лингвистическое обеспечение. Да, да, тот самый заветный словарь ключевых слов и предметных рубрик, который вы давно уже готовы сделать (опыт-то есть!), но все руки никак не доходят. Вы можете унифицировать его и держать под рукой либо в печатном виде, либо в электронном (как вам больше нравится), можете использовать его для обучения молодых сотрудников, для обслуживания читателей. Как создать его основу, описано ниже.

Второй пример. Ни для кого не секрет, что стандарты составления библиографического описания меняются, и каждый раз это процесс болезненный: надо решить, что делать с записями, составленными в соответствии со старыми стандартами. Оказывается, используя описываемую технологию, вы можете решить и эту проблему. Как это сделать, будет рассказано в следующей статье.

Список этот далеко не исчерпан. Авторы приглашают коллег для постановки новых задач, так как возможности технологии достаточно велики.

Алгоритм

Технология опробована на базе данных «Электронный каталог статей по теме», ведется 4 года, объем – 11 тысяч записей, использована СУБД CDS/ISIS.

Хочется сказать отдельно об особенностях технологии.

Программа AWK предназначена для обработки текстовых файлов (см. статью: AWK - универсальная программа работы с текстовыми файлами/ Бочаров Б.П., Воеводина М.Ю.// Библиотеки учебных заведений, 2002 , № 4, с.39-53). В связи с этим работа ведется в несколько этапов по следующему алгоритму.

- Импортируем из БД содержимое записей либо всех, либо для нужного диапазона номеров записей. Получаем ISO – файл (стандарт ISO 2709). Эту операцию выполняет сама СУБД. Полученный файл разбит на строки по 80 символов.

- С помощью программы **del_cr.awk** (см. Приложение) убираем разбиение на сторки. Получаем полноценный ISO – файл.

- С помощью программы **iso2text.awk** (см. Приложение) преобразуем ISO – файл в текстовый (TXT). В этом файле каждая запись будет представлена в таком виде:

  первая строка - маркер записи;

  последующие строки – поля, каждое поле - на новой строке, каждый экземпляр поля тоже на новой строке;

  подполя разделяются знаком «^» с последующим кодом подполя.

  В последней строке приводится номер записи в БД. Именно этот текстовый файл является основным объектом в нашей работе. Именно его мы будем обрабатывать различными программами при глобальной корректировке базы данных.

- Теперь, обрабатывая текстовый файл с помощью соответствующих программ, можно выполнить намеченное. Если вы занимаетесь только поиском ошибок, то получаем файл с перечнем ошибочных записей и исправляем их вручную. Если вы выполняете преобразование текстового файла, переходим к следующему пункту.

- Когда мы сделали со своим текстовым файлом все, что хотели, надо вставить его на место. Для этого из текстового он должен снова стать ISO – файлом. Выполняем это с помощью программы **text2iso.awk** (см. Приложение).
- И последнее. Выполняем импорт нашего улучшенного корректировкой ISO – файла БД. Напоминаем, что импорт должен производиться заменой, а не добавлением! Эту операцию выполняет сама СУБД.

С чего начать?

Предлагаем с простого. В каждой БД есть контролируемые поля, которые никогда не могут быть пустыми. Например, для нашей БД - это поля «Источник», «Каталогизатор» и др.

На первом этапе работы хотелось бы знать, в каких записях эти поля не заполнены. Для этого используем программу **pust_015.awk,** которая из полученного текстового файла «вытащит» для нас номера записей, которые надо исправить.

Текст программы приводится в каталоге **part0** приложения. В тексте можно видеть регулярное выражение /^015/ - это соответствует полю «Источник» для нашей базы данных. На выходе формируется текстовый файл, где перечислены номера записей, в которых отсутствует поле 015. Остается только внести исправления - зная номера записей, это можно сделать вручную.

Поменяв регулярное выражение на /^082/, получим аналогичный список для поля «Каталогизатор». Подобным образом можно осуществить проверку всех интересующих вас полей. Затраты на такую операцию несравнимы с прямым просмотром !

Второй по сложности этап – это нахождение в полях и подполях недопустимых символов. Например, в поле «Индекс ББК» не может быть знака «+», наличие же его в этом поле говорит о том, что поле заполнено неправильно и поиск будет вестись некорректно.

Программа **bbk_pl.awk**, текст которой приведен в каталоге **part0** приложения, сформирует текстовый файл, в котором перечислены номера неправильно заполненных записей. Аналогично можно обрабатывать другие поля, делая замены в соответствующих регулярных выражениях.

Ну, и, наконец-то, самое интересное. Наиболее сложный и впечатляющий этап вашей работы - это автоматическая вставка скорректированного поля или подполя. С помощью программы **get_fld.awk** (см. Приложение) получаем текстовый файл с содержимым поля (или подполя).

Какого? В нашем примере - это поле 078 «Ключевые слова» и подполе 211^c «Инициалы автора». Это также может быть поле 005 «Предметные рубрики», 009 «Индекс ББК» или какое-либо другое.

Теперь нужно произвести корректировку. Для этого есть несколько способов.

Авторы считают удобным воспользоваться таблицей WORD. В одну колонку таблицы помещаем содержимое извлеченных полей. Во второй колонке должны появиться соответствующие новые значения полей. Естественно, то, что не подлежит корректировке, просто копируем. Таким образом, каждому извлеченному из БД значению ставим в соответствие исправленное либо прежнее значение.

Таблицу преобразовываем в текстовый файл. Все готово для автоматической вставки новых значений поля. Осуществляется она программой **rpl_fld.awk** (см. Приложение).

На основе исправленного списка содержимого поля «Ключевые слова» можно начинать формировать основу своего лингвистического обеспечения. Впрочем, это уже совсем иная задача, которая остается за пределами данной статьи.

ПРИЛОЖЕНИЕ

На сайте журнала “Библиотеки учебных заведений” представлены все программы, реализующие предложенную технологию. Ниже подробно описан алгоритм их работы.

Шаг 1. Перевод файла, экспортированного из ISIS в текстовый вид.

В каталоге **part1** приложения находятся следующие файлы:

src.iso	- файл, экспортированный из ISIS,
del_cr.awk	- программа, удаления лишних концов строк,
iso2text.awk	- перевод iso-файлов в текстовый вид,
i2t.bat	- вызов программ del_cr.awk и iso2text.awk.

Рассмотрим подробно, как работают эти программы. Для выполнения операции перевода файла, экспортированного из ISIS в текстовый вид необходимо выполнить следующий командный файл:

```
rem      i2t.bat
rem Перевод файла, экспортированного из ISIS в текстовый вид.
..\gawk --re-interval -f del_cr.awk src.iso >cor.iso
..\gawk --re-interval -f iso2text.awk cor.iso >cor.txt
```

Сначала выполняется программа, исправляющая изуродованный ISIS’ом iso-файл.

```
#      del_cr.awk
# Удаление лишних концов строк
                    { printf("%s",$0); }
```

В результате работы этой программы появляется файл промежуточный файл cor.iso.

Затем вызывается программа, перевода iso-файла в текстовый вид.

Напомним, что запись в формате ISO 2709 начинается с маркера (24 символа). Затем идет справочник, который состоит из нескольких статей, длиной 12 символов (по количеству полей в записи). Первые три символа в статье – номер поля. В конце справочника ставится символ - разделитель полей. Дальше записывается содержание полей, разделенных на подполя. После каждого поля ставится разделитель полей. В конце записи ставится символ – разделитель записей.

Мы собираемся записывать измененную информацию в iso-файл, поэтому нужно сохранить в маркере все поля, не относящиеся к длине записи и адресам данных внутри записи. В маркере первые 5 символов - длина записи, а 5 символов, начиная с 13-го – базовый адрес данных (смещение 1-го подполя).

Текст программы:

```
#       iso2text.awk
# Перевод iso-файлов в текстовый вид
BEGIN{
# Устанавливаем разделитель подполей
  FS = "$";
# Устанавливаем разделитель записей
  RS = "#";
}

{
# Записываем маркер и справочник в переменную tmp
  tmp = $1;
# Записываем в переменные m1 и m2 ту часть маркера,
# которая не относится к длине записи
  m1 = substr(tmp,6,7);
  m2 = substr(tmp,18,7);
# Записываем маркер в файл.
```

```
# Вместо длины записи и базового адреса выводим xxxxx
  printf("xxxxx%sxxxxx%s\n",m1,m2);
# Удаляем маркер из временной переменной
  sub(/^.{24}/,"",tmp);
  i = 0;
# Записываем в цикле номера полей в массив nfld
  while(tmp != "){
    nfld[i++] = substr(tmp,1,3);
    sub(/^.{12}/,"",tmp);
  }
# Выводим номера и содержимое (если не пустое) полей
  for(i=2; i<=NF; i++)
    if($i != "") print nfld[i-2] $i;
# Выводим строку, обозначающую конец записи
  print "***** End of record N " NR;
}
```

В результате работы программы появляется файл cor.txt. Содержимое этого файла:

```
xxxxx0000000xxxxx0004500
211^aШрайберг^cЯ.Л.
215^aОсторожно: автоматизация и рядом Интернет! Не
носите розовых очков^eПроблемный доклад
015^aНауч. и техн. б-ки
016^h1997
025^aN 1
027^aC.53-64
009^a78.305.8^c73.8
075^a78.305.8
005^bАвтоматизация библиотек^dИнтернет
078^aбиблиотеки^bсистемы автоматизированные
информационные^cпочты электронные^dтелекоммуникации^ea
втоматизация^fпрограммное обеспечение^iCD-
```

```
ROM^jИнтернет^gCDS/ISIS^hхозцентры
082^a1
080^b312
999^a06-14-00  15:16:20
***** End of record N 1
xxxxx0000000xxxxx0004500
211^aШрайберг^cЯ.
215^aCDS/ISIS и международное сотрудничество:
отечественный опыт
015^aБиблиотека
016^h1999
025^aN 1
027^aC. 49-51
009^a32
075^a32
078^aпрограммное обеспечение^bпакет прикладных
программ^cCDS/ISIS
082^a1
080^b312
999^a03-17-99  13:16:48
***** End of record N 2
xxxxx0000000xxxxx0004500
211^aШрайберг^cЯ.Л.
215^aАвтоматизация библиотек сегодня: оценка и
осмысление подходов и проблем
015^aНауч. и техн. б-ки
016^h1999
025^aN 2
027^aC. 4-18
009^a78.305.8
075^a78.305.8
078^aАвтоматизация библиотек^bинформационные технологи
и^cинформатизация^dпонятие^eтерминология^fСУБД^iПоиск
```

```
овые системы^jCDS/ISIS^gППП^hпрограммное обеспечение
078^aИнтернет
082^a1
080^b312
999^a01-12-00  14:50:20
***** End of record N 3
```

Шаг 2. Выбор и анализ содержимого конкретного поля.

В каталоге **part2** приложения находятся следующие файлы:

cor.txt	- файл, полученный на предыдущем шаге,
get_ fld.awk	- программа, выбирающая из текстового файла значения конкретных полей (под полей),
go.bat	- вызов программ get_fld.awk.

Мы будем использовать программу, позволяющую вывести значение любого поля (подполя).

Вызов этой программы осуществляется с помощью командного файла **go.bat** следующим образом:

go XXXa,

XXX – номер поля (ведущие нули обязательны, например 005),

a – буква подполя (может быть опущена).

Имя выходного файла: **XXXa.val**.

Примеры вызова:

Вызов	Выполняемое действие	Выходной файл
go 211a	Выбор фамилий авторов	211a.val
go 078	Выбор ключевых слов	078.val

Текст файла go.bat:

```
rem     go.bat
rem  Вызов программы выбора значений поля
..\gawk --re-interval -f get_fld.awk -v fld=%1 cor.txt
>%1.zzz
sort %1.zzz > %1.val
del %1.zzz
```

Сначала мы записываем значения полей в промежуточный файл XXXa.zzz, затем сортируем этот файл. Результат сортировки – файл XXXa.val. В конце работы мы удаляем промежуточный файл.

Комментарии в тексте программы get_fld.awk подробно описывают ее работу.

```
#     get_fld.awk
# Вывод значений поля (подполя)
BEGIN{
# Устанавливаем разделитель подполей
  FS = "^";
# Обрабатываем параметр fld - код поля (подполя)
# Ошибки, связанные с неправильным вводом, не
обрабатываются
# cd - код поля (три цифры)
  cd = fld;
# удаляем из строки все символы,
# не являющиеся цифрами
```

```
  gsub(/[^0-9]/,"",cd);
# sf - буква подполя (sf может быть пустой строкой)
  sf = fld;
# удаляем из строки все цифры
  gsub(/[0-9]/,"",sf);
}

{
# Если код поля не совпадает, переходим к следующей строке
  if($1 != cd) next;
# Код поля совпал, обрабатываем подполя
  for(i=2; i<=NF; i++){
# Определяем букву (let) и значение (val) текущего подполя
    val = $i;
# Выбираем из строки 1-й символ
    let = substr(val,1,1);
# Удаляем из строки 1-й символ
    sub(/^./,"",val);
# Если задана буква подполя и буква текущего подполя
# с ней не совпадает, переходим к следующему подполю
    if((sf != "") && (let != sf)) continue;
# Увеличиваем количество повторений для значения подполя
    arr[val]++;
  }
}

END{
# Выберите нужные операторы вывода,
# ненужные закомментируйте
# Вывод значений подполя и числа повторений
#  for(i in arr) printf("%s\t\%d\n",i,arr[i]);
# Вывод только значений подполя
#  for(i in arr) printf("%s\n",i);
```

```
# Вывод только значений подполя 2 раза
# для преобразования в таблицу Word
  for(i in arr) printf("%s\n",i);
  for(i in arr) printf("%s\n",i);
}
```

Применим эту программу для исправления ошибок в исходном файле.

Сначала проверим поле 211c – инициалы автора. С помощью команды go 211c выведем значения этого поля в файл 211c.val. Заметим, что в программе get_fld.awk мы два раза выводим одни и те же значения. Это сделано для удобства проверки информации в Word.

Переводим значения поля 211c в таблицу Word:

1. В текстовом редакторе FAR копируем содержимое файла 211c.val в буфер обмена.

2. Вызываем Word и создаем файл 211c.doc.

3. Вставляем в этот файл информацию из буфера обмена.

4. Выделяем всю вставленную информацию и вызываем пункты меню «Таблица – Преобразовать – Текст в таблицу». В появившемся окне указываем «количество колонок» - 2.

Информацию в левом столбце оставляем без изменений, а в правом столбце исправляем ошибки. В данном случае в первой строке таблицы (ошибочное значение подчеркнуто) добавляем пропущенное отчество в инициалах автора. У нас получилась такая таблица:

Я.	Я.Л.
Я.Л.	Я.Л.

Заметим, что программы FAR и Word автоматически пере-

кодировали текст из кодировки DOS в Windows.

Для дальнейшего изменения информации в iso-файле нам необходимо перевести эту таблицу в текстовый вид и произвести обратную перекодировку из Windows в DOS.

Алгоритм действий такой:

1. Сохраняем файл 211c.doc в текстовом виде (меню «Файл – Сохранить как», тип сохраняемого документа «только текст»).

2. Закрываем Word и вызываем на редактирование в FAR файл 211c.txt (только что полученный из Word). Копируем содержимое файла в буфер обмена. Выходим из редактора.

3. Создаем новый файл 211c.rpl. Устанавливаем его кодировку как DOS. Вставляем информацию из буфера обмена.

4. Удаляем все пустые строки в конце файла.

Файл 211c.rpl будет использован на этапе замены информации в текстовом файле.

Аналогичным образом получаем файл 078.rpl – файл ключевых слов (все подполя поля 078).

Сразу видно, что в названии CDS/ISIS хотя бы один раз была допущена ошибка (первая буква не латинская, а русская). Фрагмент файла 078.doc представлен ниже, ошибочное значение подчеркнуто.

программное обеспечение	программное обеспечение
CDS/ISIS	CDS/ISIS
системы автоматизированные информационные	системы автоматизированные информационные

Шаг 3. Замена в текстовом файле.

В каталоге part3 приложения находятся следующие файлы:

cor.txt	- файл, полученный на шаге 1,
078.rpl	- файл замен для поля 078,
211c.rpl	- файл замен для подполя 211c,
rpl_fld.awk	- программа, заменяющая значения полей (подполей) в текстовом файле,
go.bat	- командный файл, осуществляющий последовательную корректировку подполя 211c и поля 078.

Вызов программы rpl_fld.awk аналогичен вызову программы get_fld.awk, описанной выше.

Имя входного файла замен XXXa.rpl (см. шаг 2).

Если значение подполя не найдено в файле замен, то информация об этом подполе выводится в файл ошибок XXXa.err.

Текст программы приведен ниже, комментарии подробно описывают ее работу.

```
#    rpl_fld.awk
# Замена значений поля (подполя)
BEGIN{
# Устанавливаем разделитель подполей
  FS = "^";
# Обрабатываем параметр fld - код поля (подполя)
# Ошибки, связанные с неправильным вводом, не
# обрабатываются
# cd - код поля (три цифры)
  cd = fld;
# удаляем из строки все символы,
# не являющиеся цифрами
  gsub(/[^0-9]/,"",cd);
```

```
# sf - буква подполя (sf может быть пустой строкой)
  sf = fld;
# удаляем из строки все цифры
  gsub(/[0-9]/,"",sf);
# Формируем имена файлов
# fn_rpl - файл замен
  fn_rpl = fld ".rpl";
# fn_err - файл ошибок
  fn_err = fld ".err";
# Читаем файл замен
# В массиве rpl_arr:
#  индексы элементов  - старая информация
#  значения элементов - новая информация
  while (getline tmp0 < fn_rpl){
    getline tmp1 < fn_rpl;
    rpl_arr[tmp0] = tmp1;
  }
# iso_nr - номер записи в iso-файле
  iso_nr = 1;
}

# Если строка начинается с 5-и звездочек,
# то достигнут конец записи.
/^\*\*\*\*\*/{
# Увеличиваем номер записи на 1
  iso_nr++;
}

{
# Если код поля не совпадает, записываем строку
# без изменений и переходим к следующей строке
  if($1 != cd){
    print $0;
```

```
    next;
  }
# Код поля совпал, обрабатываем подполя
# tmp - временная переменная для записи измененной строки
  tmp = cd;
  for(i=2; i<=NF; i++){
# Определяем букву (let) и значение (val) текущего подполя
    val = $i;
# Выбираем из строки 1-й символ
    let = substr(val,1,1);
# Удаляем из строки 1-й символ
    sub(/^./,"",val);
# Если задана буква подполя и буква текущего подполя
# с ней не совпадает, записываем подполе без
# изменений в переменную tmp
# и переходим к следующему подполю
    if((sf != "") && (let != sf)){
      tmp = tmp "^" $i;
      continue;
    }
# Если значение подполя не найдено в массиве
# подстановок, записываем подполе без
# изменений в переменную tmp, выводим
# информацию о подполе в файл ошибок
# и переходим к следующему подполю
    if(rpl_arr[val] == ""){
      tmp = tmp "^" $i;
      printf("%5.5d [%s] :%s\n",iso_nr,let,val) >fn_err;
      continue;
    }
# Значение подполя найдено в массиве
# подстановок, записываем новое значение
# подполя в переменную tmp
```

```
   tmp = tmp "^" let rpl_arr[val];
 }# конец цикла for(i=2; i<=NF; i++)
# Выводим номер и значение поля
  print tmp;
}
```

Командный файл go.bat последовательно вызывает программу rpl_fld.awk для замены значений подполя 211c и поля 078. После изменения поля 211c информация записывается в файл tmp.zzz, который является входным для замены поля 078. Затем этот временный файл удаляется.

В результате получается файл **new.txt.**

```
rem     go.bat
rem   Вызов программы замены значений поля

..\gawk --re-interval -f rpl_fld.awk -v fld=211c cor.txt
>tmp.zzz
..\gawk --re-interval -f rpl_fld.awk -v fld=078 tmp.zzz
>new.txt
del *.zzz
```

Шаг 4. Преобразование текстовых файлов в iso-файлы.

В каталоге part4 приложения находятся следующие файлы:

new.txt	- текстовый файл, который нужно преобразовать, в iso-файл.
text2iso.awk	- преобразование текстовых файлов в iso-файлы,
t2i.bat	- вызов программы text2iso.awk.

При преобразовании текстового файла в iso-файл нужно решить следующие задачи:

1. Записать в справочник номера полей, содержание полей сохранить отдельно.

2. Определить и записать числовую информацию в каждую статью справочника. За номером поля (3 символа) должны следовать длина содержания поля (4 символа) и позиция начального символа (5 символов).

3. Заполнить в маркере длину записи (первые 5 символов) и базовый адрес данных (5 символов, начиная с 13-го).

4. Записать в файл маркер, справочник и содержание полей.

Все эти задачи решает программа **text2iso.awk.**

```
#    text2iso.awk
# Преобразование текстовых файлов в iso-файлы
BEGIN{
# В отдельные переменные записываем
# разделитель полей (iso_FS) и
# разделитель записей (iso_RS) в iso-файле
  iso_FS = "$";
  iso_RS = "#";
}
#####################################
# В 1-й строке текстового файла – маркер,
# начинается с символов xxxxx
#####################################
/^xxxxx/{
# Переменная curNfld будет использована для
# формирования массивов номеров
# и содержания полей
  curNfld = 0;
# Запоминаем в переменных m1 и m2
# всю информацию, которую не будем изменять
  m1 = substr($0,6,7);
  m2 = substr($0,18,7);
```

```
# Переходим к следующей строке
  next;
}
#######################################
# Конец записи в текстовом файле. Формируем
# запись для iso-файла и выводим ее
#######################################
/^\*\*\*\*\*/{
# spr – временная переменная для справочника
  spr = "";
# rec – временная переменная для
# содержания полей
  rec = "";
# adr – временная переменная для позиции начального
# символа содержания полей
  adr = 0;
# Цикл по всем полям
  for(i=0; i<curNfld; i++){
# l – длина содержимого поля
    l = length(fld_val[i]);
# В справочник записываем текущую статью
    spr = spr sprintf("%3.3d%4.4d%5.5d",fld_
num[i],l,adr);
# Во временную переменную добавляем
# содержание текущего поля
    rec = rec fld_val[i];
# Увеличиваем позицию начального символа на
# длину поля (для следующего поля)
    adr += l;
  }
# Добавляем к справочнику разделитель полей
  spr = spr iso_FS;
# Добавляем к содержимому полей разделитель записей
```

```
  rec = rec iso_RS;
# Базовый адрес данных – длина справочника плюс
# длина маркера (24)
  adr = length(spr)+24;
# Длина записи – длина справочника плюс длина
# содержимого полей плюс длина маркера (24)
  l = length(spr)+length(rec)+24;
# Выводим запись – маркер (l,m1,adr,m2), справочник
# и содержание полей
  printf("%5.5d%s%5.5d%s%s%s",l,m1,adr,m2,spr,rec);
# Удаляем массивы
  delete fld_val;
  delete fld_num;
# Переходим к следующей строке
  next;
}
#####################################
# Строка, содержащая номер и
# содержание поля
#####################################
{
# vfl - содержание поля, заносим туда все
# символы из строки, за исключением 1-х трех
  vfl = $0 iso_FS;
# nfl - номер поля, заносим туда 1-е 3 символа
  nfl = substr(vfl,1,3);
  sub(/^.{3}/,"",vfl);
# Заносим информацию в массивы:
# fld_num - номера полей,
# fld_val – значения полей.
  fld_num[curNfld] = nfl;
  fld_val[curNfld] = vfl;
# Увеличиваем текущий индекс массивов на 1
  curNfld++;
}
```