ТАРТУСКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

ТРУДЫ

ВЫЧИСЛИТЕЛЬНОГО ЦЕНТРА

49

ТАРТУ
1982

ТАРТУСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМЫ ОБРАБОТКИ ДАННЫХ

ТРУДЫ ВЫЧИСЛИТЕЛЬНОГО ЦЕНТРА

ВЫПУСК 49

ТАРТУ 1982

СИСТЕМЫ ОБРАБОТКИ ДАННЫХ.
Труды вычислительного центра. Выпуск 49.
На русском языке.
Тартуский государственный университет.
ЭССР, 202400, г.Тарту, ул.Юликооли, 18.
Ответственный редактор К. Наремаа.
Корректор А. Райд.
Подписано к печати 23.11.1982.
МВ 09650.
Формат 60x84/16.
Бумага писчая.
Машинопись. Ротапринт.
Условно-печатных листов 6,51.
Учетно-издательских листов 4,42.
Печатных листов 7,0.
Тираж 200.
Заказ № 1222.
Цена 65 коп.
Типография ТГУ, ЭССР, 202400, г.Тарту, ул.Пяэкооли, 14.

СОВМЕСТНОЕ ПОЛЬЗОВАНИЕ ДАННЫМИ В СИСТЕМЕ РАМА

Ю. Каазик, М. Томбак

Управление системой РАМА (см. [1]) проектировано с учетом возможности одновременного обслуживания многих пользователей, каждый из которых может иметь по несколько программ. Такая совместная работа пользователей может быть реализована как в пакетном режиме, так и в режиме разделения времени. В обоих случаях, однако, пользователи обслуживаются параллельно, а их программы последовательно: активной программой у каждого пользователя всегда является не более одной.

Программами пользователей системы РАМА можно считать и заказы на транслирование легенды (легенда как средство определения структуры записи описана в [2] и [6]), описания файла (см. [3]) или конкретной программы обработки. Типичная же программа пользователя, это заказ на выполнение оттранслированной уже программы для создания, обработки или использования информации банка данных. Такими программами могут быть, например, DDL-программы (см. [4] и [8]) или же DML-программы (см. [5] и [9]).

Одновременной работой активных программ разных пользователей и обрабатываемой ими информацией управляет (при помощи ОС) специальный монитор системы РАМА. Основные проблемы та-

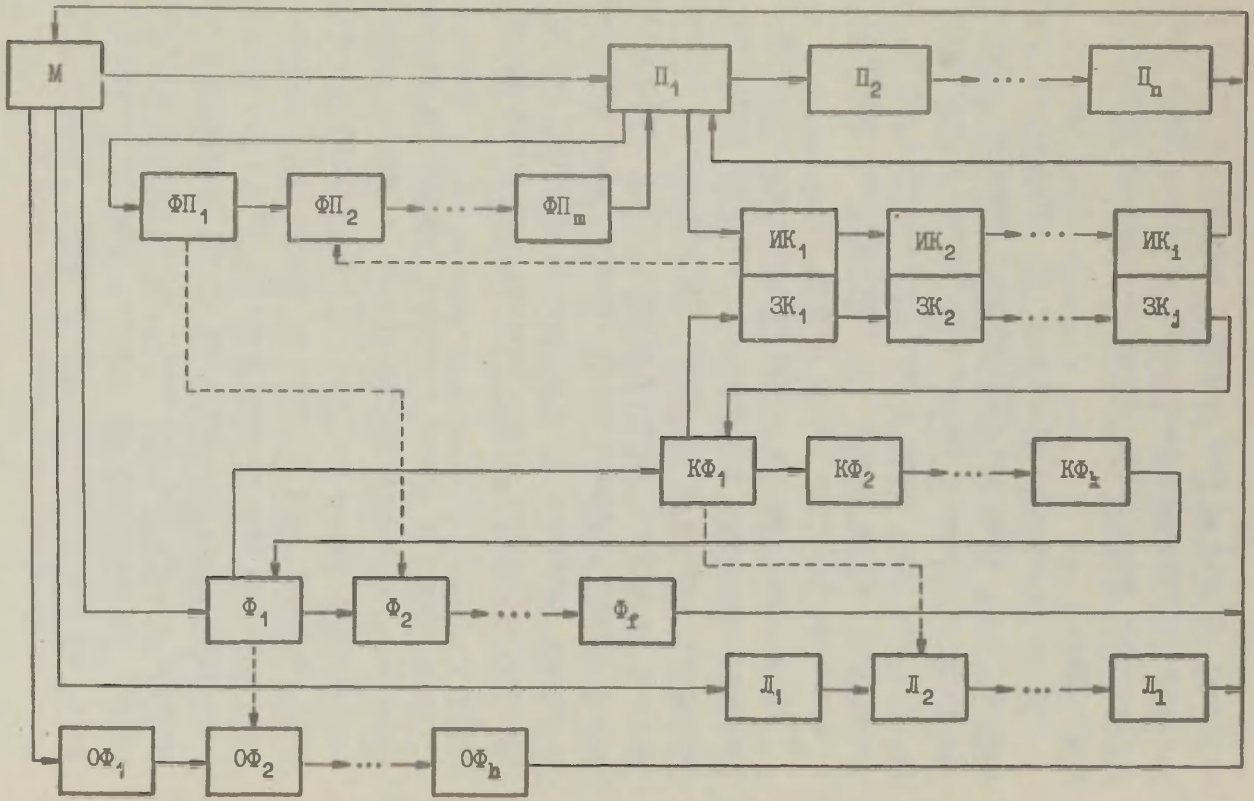
кого управления связаны со случаем, когда активными программами пользователей являются программы типа DML-программ, обрабатывающие записи из одних и тех же файлов. В дальнейшем изложении мы и будем иметь в виду именно этот случай.

При управлении одновременной работой нескольких программ над одними и теми же файлами монитор должен оперировать многими различными объектами - программами пользователей, файлами, их описаниями, записями и т.п. Такое оперирование означает как быстрое нахождение требуемых объектов, так и ввод новых, своевременное удаление лишних или же определение связей между разными объектами. Естественно требовать, чтобы необходимые объекты при этом вводились по возможности только в одном экземпляре.

Для ускорения и систематизации такого управления основные характеристики каждого объекта выделены в виде специального поля. Эти поля, как узлы или звенья соответствующими ссылками объеденены в циклические списки (цепочки), общая картина которых схематически представлена на стр. 5.

Непосредственно монитору (М) подчиняются четыре списка: список имеющихся пользователей (П), список открытых для пользователей файлов (Ф), список описаний этих файлов (ОФ) и список всех необходимых легенд (Л). На рисунке число пользователей обозначено через n, число файлов через f, число их описаний через h (очевидно $h \leq f$) и число легенд через l.

Каждому узлу списка П подчинены еще два списка: список файлов этого пользователя (ФП) и список имен комплектов (ИК), используемых в активной программе этого пользователя (комплект - это множество записей одного файла, описанных одной


М
П1
П2
Пn
ФП1
ФП2
ФПm
ИК1
ИК2
ИКi
ЗК1
ЗК2
ЗКj
КФ1
КФ2
КФk
Ф1
Ф2
Фf
Д1
Д2
Дl
ОФ1
ОФ2
ОФh

легендой). Каждому же узлу списка Ф подчинен еще список используемых в программах комплектов этого файла (КФ) - максимальная длина этого списка определена описанием соответствующего файла.

Под каждым узлом каждого списка КФ строится еще список записей этого комплекта (ЗК), причем в качестве узлов используются не отдельные поля, а узлы списков ИК всех пользователей. Эти узлы, таким образом, объеденены в списки при помощи двух систем ссылок - по пользователям (списки ИК) и по комплектам различных файлов (списки ЗК).

Такая двойная система ссылок позволяет при обращении какой-нибудь программы пользователя к очередной записи быстро установить, введена ли эта запись уже для другого пользователя или для другого имени комплекта того же пользователя. Ведь из каждого комплекта любого открытого файла можно ввести столько различных записей, сколько различных имен этого комплекта встречается во всех активных программах пользователей.

Связи между различными списками реализованы ссылками, некоторые представители которых указаны на рисунке пунктиром. Так, например, каждый узел в списке ФП содержит ссылку на соответствующий узел в списке Ф, каждый файл в списке Ф имеет ссылку на свое описание в списке ОФ и т.д. Ссылаемые таким образом узлы содержат кроме другой информации и число входящих ссылок. Если это число равно нулю, то узел можно удалить. Например, если при удалении файла из списка Ф окажется, что число ссылок на соответствующее описание станет нулем, то удаляется и это описание из списка ОФ.

Узлы списков ФП и КФ можно считать теми объектами, которых они представляют - кроме общих характеристик они содержат только ссылки на узлы других списков. Во всех остальных списках узлы содержат еще и ссылку на представляемый объект (на рисунке такие ссылки не указаны). Так, например, узел списка Ф ссылает на блок DCB этого файла, узел списка Л на (транслированную) легенду, узел списка ОФ на описание файла и узел списка П на активную программу. Во всех этих случаях соответствие между узлом и объектом взаимно-однозначное (если активные программы двух пользователей совпадают, то эта программа представлена в двух экземплярах).

В случае списков ИК или ЗК дело сложнее: несколько узлов могут иметь ссылки на одну и ту же физическую запись (см. [7]), а в то же время под некоторыми узлами запись может отсутствовать. Ссылка на физическую запись отсутствует тогда, когда соответствующее имя комплекта в работающей части программы не используется. Одинаковые ссылки в нескольких узлах означают, что несколько пользователей или же несколько имен комплектов одной программы одновременно обращаются к одной и той же записи. Учитывая эту возможность, в физической записи запоминается число ссылающих на нее узлов списков ИК (или ЗК) - если это число станет нулем, то запись выводится.

Одновременное обращение разных пользователей к записям из одних и тех же файлов требует некоторую осторожность при вводе изменений в эти файлы. Поэтому, пользователь должен для своей программы точно указать, в каких пределах эта программа допускает совместное использование файлов. Такие указания содержатся в заказе к запуску программы, где задается

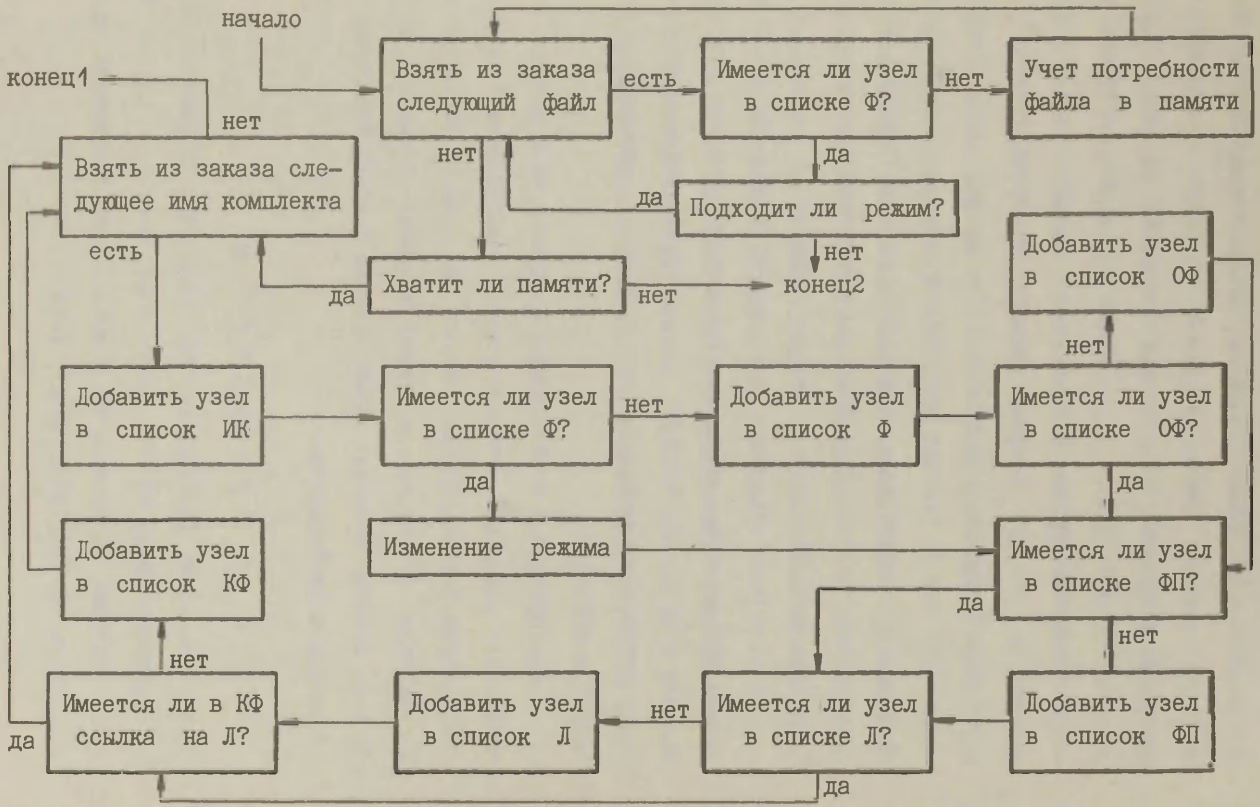
связь всех имен комплектов с конкретными файлами и фиксируется режим использования каждого файла (язык оформления таких заказов будет подробно описан в следующем выпуске Трудов ВЦ ТГУ).

В системе РАМА пользователь может выбрать один из следующих шести режимов пользования файлами (см. также [10], стр. 312): чтение (Ч), запись (З), защищенное чтение (ЗЧ), защищенная запись (ЗЗ), монопольное чтение (МЧ) и монопольная запись (МЗ). Правила совместимости режимов указаны в следующей таблице:

	Ч	З	ЗЧ	ЗЗ	МЧ	МЗ
Ч	+	+	+	+		
З	+	+				
ЗЧ	+		+			
ЗЗ	+					
МЧ						
МЗ						

Например, когда один пользователь уже обрабатывает некоторый файл в режиме ЗЧ, то следующий пользователь получает доступ к этому файлу лишь в том случае, если он указал в заказе режим Ч или ЗЧ.

Совместимость заказанных разными пользователями режимов проверяется перед открытием к запуску очередной программы пользователя. На стр. 9 приведена принципиальная блок-схема той подпрограммы монитора, которая управляет таким запуском. Эта же подпрограмма занимается созданием и преобразованием описанных выше списков.


начало
конец1
Взять из заказа следующий файл
есть
Имеется ли узел в списке Ф?
нет
Учет потребности файла в памяти
нет
да
Взять из заказа следующее имя комплекта
нет
да
Подходит ли режим?
нет
есть
да
Хватит ли памяти?
нет
конец2
Добавить узел в список ОФ
нет
Добавить узел в список ИК
Имеется ли узел в списке Ф?
нет
Добавить узел в список Ф
Имеется ли узел в списке ОФ?
да
да
Добавить узел в список КФ
Изменение режима
Имеется ли узел в списке ФП?
да
нет
нет
Имеется ли в КФ ссылка на Л?
да
Добавить узел в список Л
нет
Имеется ли узел в списке Л?
Добавить узел в список ФП
да

Начальная (верхняя правая) часть блок-схемы изучает возможность запуска рассматриваемой программы пользователя. Программа не может быть в данной ситуации запущена тогда, когда некоторый файл, заказываемый для этой программы, уже обрабатывается другим пользователем и режимы не допускают совместную работу. Вторая причина невозможности запуска программы возникает в том случае, если по предварительной оценке для нее не хватает оперативной памяти. В этих случаях подпрограмма заканчивает свою работу выходом "конец2" и рассматриваемая программа пользователя ставится в конец очереди пассивных программ.

Если запуск программы оказывается возможным, то описываемая подпрограмма монитора создает необходимые узлы всех списков и на выходе "конец1" запускает рассматриваемую программу пользователя, т. е. превращает эту программу из пассивной в активную.

Когда программа пользователя совершает свою работу (нормально или аварийно), то аналогичная подпрограмма монитора ликвидирует соответствующие списки ИК и ФП, а также удаляет лишние узлы (в случае появления таковых) в списках Ф, КФ, ОФ и Л. Память, занятая этими узлами и представляемыми ими объектами, освобождается.

Литература

1. Ю. Каазик, М. Томбак, Система РАМА для управления базой данных. Труды ВЦ ТГУ, 1978, № 41, 3-6.
2. А. Изотамм, Ю. Каазик, М. Томбак, Язык определения записи. Труды ВЦ ТГУ, 1978, № 41, 7-64.

3. Ю. Каазик, Образование файлов. Труды ВЦ ТГУ, 1978, № 41, 65-74.

4. Ю. Каазик, П. Ээльма, Ввод и корректировка данных. Труды ВЦ ТГУ, 1978, № 41, 75-96.

5. Ю. Каазик, А. Рауп, Язык манипулирования данными. Труды ВЦ ТГУ, 1978, № 41, 97-140.

6. А. Изотамм, Дерево описания записи в системе РАМА. Труды ВЦ ТГУ, 1980, № 43, 3-35.

7. А. Изотамм, Физическое представление записи в системе РАМА. Труды ВЦ ТГУ, 1980, № 43, 36-54.

8. Ю. Каазик, А. Толпин, Реализация языка ввода данных. Труды ВЦ ТГУ, 1980, № 45, 21-35.

9. А. Рауп, Реализация языка манипулирования данными. Труды ВЦ ТГУ, 1980, № 45, 36-65.

10. Информационные системы общего назначения. М., 1975.

ОРГАНИЗАЦИЯ ФАЙЛОВ В СИСТЕМЕ РАМА

А. Рауп

1. Общее описание

Файлы данных системы РАМА являются наборами данных с прямой организацией в смысле ОС ЕС, но их внутренняя структура имеет общие черты с индексно-последовательными и библиотечными наборами данных. В дальнейшем логические записи файла (структура которых описывается RDL-легендами, см. [1]) будем называть просто записями, а физические записи - блоками. Каждый файл состоит из блоков фиксированной длины. По своему содержанию блоки делятся на блоки данных и блоки каталога. Кроме этого, самый первый блок файла, т.н. нулевой блок, имеет особое значение.

Блоки данных предназначены для хранения записей файла. В одном блоке может размещаться несколько записей и одна запись может размещаться в нескольких блоках. Часть одной записи в блоке будем называть сегментом. Если запись состоит из нескольких сегментов, то эти сегменты связаны с помощью внешних указателей в одну цепь. Для доступа к записи надо знать адрес ее первого сегмента. Записи обычно расположены в порядке их поступления и всегда существует граница между использованной и свободной частью файла - порядковый номер

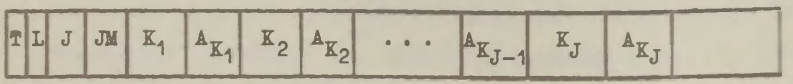
последнего заполненного блока. Новые записи добавляются всегда в конец файла и граница между использованной и свободной частями файла продвигается дальше.

Так как место нахождения записи не связано со значением ее ключа, то для доступа к записям используется каталог. Каталог файла строится в виде B-дерева (см. [2], стр. 563-570), причем каждый узел этого дерева размещается в одном блоке. Ключи в узлах B-дерева - это ключи записей, составленные по описанию файла на языке FDL (см. [3]). Кроме ключа в каталоге хранят и указатель на первый сегмент соответствующей записи. При добавлении новой записи к файлу в каталог этого файла добавляются ее ключ и адрес первого сегмента, выполняя при этом нужные преобразования B-дерева. Для удаления записи из файла достаточно удалить ее ключ и адрес из каталога. Удаление и добавление ключей в основном происходит по алгоритмам, описанным в [2], введены лишь небольшие модификации.

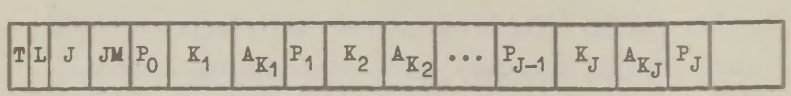
2. Каталог файла

По сравнению с B-деревом, описанным в [2], в системе РАМА введены некоторые изменения. Обозначим количество уровней B-дерева через l. На уровне l - 1 исключены пустые указатели на фиктивные листья дерева и в дальнейшем узлами уровня l мы будем считать не фиктивные листья, а узлы, содержащие только ключи и не имеющие подчиненных узлов. Напомним, что порядком B-дерева называют максимальное количество ключей, размещаемых в одном узле. Следовательно, на уровне l используется большее значение порядка чем на более высоких уровнях.

Блоки каталога, соответствующие узлам уровня 1, имеют структуру:



а блоки, соответствующие узлам высших уровней, структуру:



Обозначения полей на этих рисунках имеют следующее содержание:

T – байт признаков;

L – длина ключа в байтах;

J – текущее число ключей в блоке;

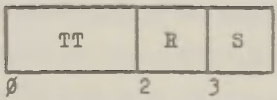
JM – максимальное возможное число ключей в блоке;

K_i – ключ (всегда $K_i < K_{i+1}$);

A_{K_i} – указатель на первый сегмент записи с ключом K_i;

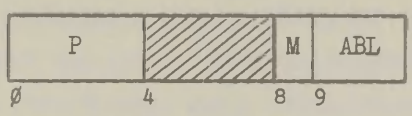
P_i – указатель на блок, соответствующий узлу следующего уровня и содержащий ключи K такие, что $K_{i-1} < K < K_i$.

Указатели (или адреса) A_{K_i} и P_i являются внешними указателями длиной в 4 байта:



где TT – порядковый номер дорожки диска, считая с начала файла, R – порядковый номер блока на дорожке, S – порядковый номер сегмента в блоке (у указателей P_i всегда S = Ø). Такую же внутреннюю структуру мы имеем в виду и в дальнейшем, если речь идет о внешних указателях и адресах.

Для ускорения обработки каталога в оперативной памяти полезно хранить как можно большую его часть. Система РАМА в оперативной памяти хранит одновременно до 1+2 блоков каталога. Данные об этих блоках собраны в таблицу каталога, которая имеется для каждого файла, находящегося в обработке. Таблица каталога состоит из 1+2 строк, каждая строка имеет длину в 12 байтов:



где P – адрес блока в файле, M – признак модификации, ABL – абсолютный адрес блока в оперативной памяти.

Первая строка таблицы соответствует всегда корню каталога, который находится во внутренней памяти в течении всей работы. В общем случае первые 1 строк таблицы хранят информацию об одном пути в каталоге от корня до уровня 1. При поиске ключа в каталоге мы спускаемся с одного уровня B-дерева на другой, и если окажется, что надо ввести блок i-того уровня, но i-тая строка таблицы занята другим блоком этого уровня, то все строки таблицы, начиная с i-того, сдвигаются на одну строку дальше. При этом блок, соответствующий последней строке, заменяется в оперативной памяти нужным блоком i-того уровня и i-тая строка таблицы заполняется. Если этот старый вытесняемый блок подвергался изменению во время нахождения во внутренней памяти, то его записывают обратно на диск. Если требуемый блок уже находится где-то в конце таблицы, то все ранее считанные блоки каталога остаются на своих местах и две строки таблицы просто обмениваются.

После добавления новой записи в конец файла ее ключ и указатель на первый сегмент записи добавляются к подходящему блоку уровня 1. Когда при этом имеет место переполнение, т.е. J > JM, то просматриваются левые и правые соседи этого блока по каталогу. Если хотя бы в одном из них еще имеется свободное пространство (J < JM), то из переполнившегося блока переносят часть ключей в соседний блок. После переноса соседние блоки заполнены всегда в равной степени. Когда оба соседа заполнены до предела (J = JM), то переполнившийся блок расщепляют на два блока и его средний ключ $K_{\frac{JM}{2}}$ добавляется к отцовскому блоку. Переполнение корня каталога вызывает всегда расщепление этого блока и создание нового корня, состоящего из одного ключа. При этом из-за увеличения на единицу числа уровней каталога увеличивается на одну строку и таблица каталога.

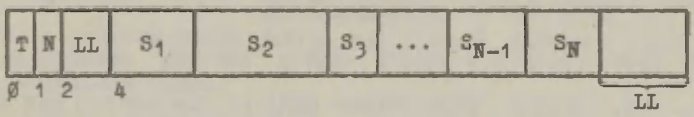
При массовом добавлении ключей в конец каталога (напр., при перезаписи файла) возникали бы повторные переносы из самого правого блока уровня 1. Во избежание таких переносов самый правый из блоков уровня 1 считается специальным и при его переполнении действуют нестандартно. Когда остальные блоки уровня 1 могут содержать от $\frac{1}{2}$ JM до JM ключей, то в последнем блоке может быть от Ø до JM ключей. При переполнении последнего блока каталога последний ключ из него (K_{JM+1}) переносят в конец отцовского блока уровня 1-1, создают новый последний блок каталога, в котором Ø ключей и куда записывают следующие ключи, добавляемые в конец каталога.

При удалении из каталога ключа, находящегося на уровне выше 1, этот ключ заменяется либо следующим, либо предыдущим

ключом из уровня 1. Потом этот ключ удаляется с 1-того уровня. Выбор предыдущего или следующего ключа зависит от того, какие блоки каталога находятся в данный момент во внутренней памяти. Когда при удалении ключа из блока уровня 1 число ключей там стало меньше чем $\frac{1}{2}$ JM, то из соседних блоков туда переносят часть ключей так, чтобы соседние блоки были одинаково заполнены. Перенос невозможен тогда, когда оба соседа имеют минимальное допустимое число ключей ($\frac{1}{2}$ JM). Тогда эти три блока объединяют в два блока, заполненные на $\frac{3}{4}$, а из их общего отцовского блока удаляется один ключ. Когда удаляется последний ключ из корня каталога, то число уровней уменьшается и соответственно уменьшают и таблицу каталога. Удаление ключа из последнего блока каталога не вызывает ни переноса ключей, ни слияния блоков.

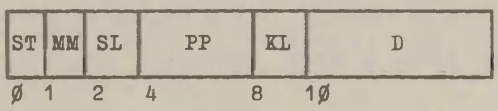
3. Блоки данных

Блоки файла, предназначенные для хранения данных (записей), имеют структуру:



где T - байт признаков, N - количество сегментов в блоке, LL - длина свободной части блока в словах, S_i - сегмент.

Каждый сегмент начинается 10-байтовым заголовком, так что сегмент имеет структуру:



где ST - байт признаков, MM - номер модификации (при первом выводе записи MM=Ø, при каждой следующей корректуре MM=MM+1), SL - длина сегмента в словах, PP - указатель на следующий сегмент данной записи или Ø, KL - общая длина записи в словах (без заголовков сегментов), D - данные (запись или ее отрывок).

При включении новой записи в файл начинают со самого последнего блока данных. Если эта запись полностью помещается в свободной части последнего блока, то она добавляется туда в качестве последнего сегмента. Если же длина записи больше чем длина свободной части блока, то в этот блок запоминается только часть записи, новым последним блоком данных считают первый свободный блок файла и там продолжают вывод оставшейся части записи. При корректировке существующих записей место включения нового экземпляра зависит от типа файла и от того, создана ли данная запись на текущем сеансе или раньше.

4. Типы файлов

Во время сеанса в каждом файле имеется фиксированное место, т.н. граница, разделяющая файл на две части. При корректировке существующих записей или блоков каталога место вывода нового экземпляра зависит от того, находится ли старый экземпляр перед или за границей. Если старый экземпляр находится за границей, то новый экземпляр записывается на место старого. В противном случае старый экземпляр сохраняется на своем месте и новый экземпляр добавляется в конец файла.

При создании нового файла определяют его тип. В системе РАМА допускаются файлы двух типов: файлы с фиксированной границей (сокращенно ФФГ) и файлы с плавающей границей (сокращенно ФПГ).

В случае ФФГ граница всегда находится перед первым блоком файла: все изменения в файле вызывают замену старых экземпляров на новые и в файле хранится только его текущее состояние. Восстановление какого-нибудь старого состояния файла невозможно.

В случае ФПГ граница находится в начале сеанса обычно перед первым свободным блоком. Вся выводимая информация записывается в конец файла и в файле сохраняются все данные, которые были в нем до начала сеанса. Сдвиг границы в конец файла в общем происходит тогда, когда заканчивают свою работу все шаги работы, использующие файл для вывода во время этого сеанса. Новые адреса корня каталога и границы для этого файла тогда заносятся в каталог файлов и каждый новый пользователь или шаг начинает свою работу уже исходя из нового состояния файла. Старые адреса корня и границы запоминаются в нулевом блоке файла, где хранят данные о всех старых состояниях файла.

Начиная работу с ФПГ, пользователь обычно исходит из самого последнего, текущего состояния файла, которое отражено в каталоге файлов. Но в заказе можно указать и на любое старое состояние файла и работать с файлом так, как-будто более поздних изменений в нем не было. Такое использование старого состояния файла разрешается только в режиме монопольного чтения (см. [4]).

Когда работа одного шага с ФПГ заканчивается аварийно, то выдается соответствующее сообщение как для этой программы, так и для других программ, параллельно записывающих в данный файл. Граница файла теперь дальше не сдвигается и в каталог файлов заносится отметка об особом состоянии этого файла. Файлов, находящихся в особом состоянии, нельзя использовать обычным путем (все такие программы пропускаются). Но если в заказе указано, что особое состояние файла известно, то работа с этим файлом начинается как бы со середины - часть информации уже находится за границей и считают, что она занесена туда во время текущего сеанса. Если теперь работа завершается успешно, то особое состояние файла отменяется, граница сдвигается и запоминается новое текущее состояние.

Особое состояние файла может быть отменено еще и администратором, который может сдвигать границу в конец файла или считать текущим состоянием для дальнейшей работы любое из старых состояний, запоминаемых в нулевом блоке файла.

Учитывая понятие границы рассмотрим несколько подробнее, как происходит обновление и удаление записей.

Если старый экземпляр записи находится перед границей, то обновленная запись помещается в конец файла так же, как и новая запись. При этом корректируется указатель на эту запись в каталоге. Но если старый экземпляр уже находится за границей, то новый экземпляр записывается на место старого. При этом сегменты записи могут увеличиваться за счет свободных пространств в блоках. Если новый экземпляр длиннее старого настолько, что не помещается и в эти увеличенные сег-

менты, то оставшаяся часть записи заносят в конец файла. Если новый экземпляр короче старого, то ненужные последние сегменты превращаются в фиктивные, т.е. в состоящие только из заголовка. Если фиктивный сегмент окажется самым последним сегментом своего блока, то он удаляется полностью.

Для удаления записи, которая находится перед границей, удаляют только из каталога файла ключ и указатель на запись. При удалении записи, находящейся за границей, кроме этого и все ее сегменты превращают в фиктивные.

С местонахождением границы считают и при внесении изменений в каталог файла. Если блок каталога уже находится за границей, то его новый экземпляр запоминается на место старого. Если же старый экземпляр находится перед границей, то новый экземпляр помещается в конец файла и обновляют указатель на этот блок в отцовском блоке каталога. Поэтому, любое изменение в каталоге влечет за собой и перенос за границу всех блоков каталога, находящихся на пути от корня до обновляемого места.

5. Эффективность использования внешней памяти

Из алгоритмов добавления и удаления ключей следует, что в среднем $\frac{3}{4}$ общего объема блоков каталога хранят полезную информацию (самих ключей и указателей на записи). Это утверждение верно, если предположить, что записи поступают в произвольном порядке. Но если записи включаются в файл в порядке возрастания их ключей, то блоки каталога заполняются максимально и неиспользованными остаются только части корня и самого последнего блока каталога.

Блоки данных заполняются максимально и при произвольном порядке поступления записей. Свободные пространства в блоках данных появляются только при обновлении (и удалении) существующих записей. При работе с ФФГ все время перевычисляется суммарный объем свободных частей блоков данных. В конце сеанса проверяют, превышает ли эта величина заданную (например, четвертую) часть общего объема всех блоков данных. Если это так, то файл автоматически сжимают. Сжатие файла состоит в его перезаписи, причем записи берутся в порядке возрастания их ключей и в конечном итоге как каталог файла, так и блоки данных будут максимально заполнены. Это стандартное поведение системы может быть отменено соответствующим заказом. Пользователь может потребовать применить сжатие в конце сеанса во всяком случае или же наоборот, запретить сжатие файла. Такое нестандартное поведение системы возможно только тогда, когда файл находится в монопольном владении пользователя.

Все сказанное о сжатии имеет место и для части ФПГ, находящейся за границей. В случае ФПГ сжатие не может охватывать весь файл, так как тогда уничтожались бы все ранние состояния файла. Заказ для сжатия всего ФПГ может быть получен только от администратора.

В случае ФПГ растрата объема внешней памяти, конечно, больше чем в случае ФФГ, так как при обновлении записей, существующих уже до начала текущего сеанса, старые экземпляры полностью сохраняются. Поэтому, время от времени придется просить администратора сжимать весь файл. Опасность переполнения файла можно уменьшить еще следующим способом. В заказе

можно потребовать, чтобы в конце сеанса ФПГ остался в особом состоянии. Работая несколько сеансов подряд с таким файлом, можно избежать ненужного дублирования записей.

Когда во время очередного сеанса все-таки происходит переполнение файла, то в случае ФФГ происходит сжатие всего файла, а при ФПГ сжимается часть, находящаяся за границей. Если переполнение файла происходит во второй раз в течение одного сеанса, то всякая дальнейшая попытка записать что-то в этот файл вызывает прекращение работы данной программы и файл сам остается в особом состоянии. Эта единственная возможность появления особого состояния у ФФГ.

6. Каталог файлов

Чтобы начать работу с файлами системы РАМА, надо знать (кроме параметров, нужных для ОС ЕС) некоторые характеристики: тип файла (ФФГ, ФПГ), адрес корня каталога, состояние файла (особое или нет) и т.д. Эти данные хранятся в двух местах - в каталоге файлов и в нулевых блоках файлов.

Каталог файлов является специальным файлом, где каждая запись состоит из основных параметров одного файла. В смысле ОС ЕС каталог файлов является библиотечным набором данных. Первая информация о файле вводится в каталог файлов тогда, когда система РАМА получает заказ для создания и принятия на учет нового файла с задаваемыми параметрами (имя и тип файла, длина блока, имя FDL-легенды и т.д.). После этого в конце каждого сеанса, где этот файл подвергался обновлению, в каталог файлов записываются данные, соответствующие текущему состоянию файла.

Записи в каталоге файлов состоят из следующих полей:

Ø	FN		
8	FDLN		
16	DO	T	BL
24	INPAR		
32	OUTPAR		
40	AR	AB	
48	AF	AD	
56	FR	DC	

,

где FN - имя файла, FDLN - имя FDL-легенды, DO - дата создания файла, T - байт признаков (тип, состояние), BL - длина блока в словах, INPAR, OUTPAR - макеты паролей для проверки полномочий пользователя, AR - адрес корня каталога, AB - адрес границы, AF - адрес первого свободного блока, AD - адрес последнего блока данных, FR - суммарная длина в словах свободных частей блоков данных, находящихся за границей, DC - дата создания текущего положения файла.

Из этих полей первые пять двойных слов остаются неизменными на все время существования файла (кроме признака особого состояния в байте признаков), а содержание последних трех двойных слов может меняться в конце каждого сеанса.

Вся информация о файле в каталоге файлов дублируется и в нулевом блоке самого файла. Это позволяет при неожиданной неудаче с каталогом файлов восстановить его содержание. Кроме этого в нулевых блоках ФПГ хранятся еще прежние состояния файлов. Для запоминания каждого состояния требуется три двойных слова, соответствующие трем последним двойным словам

в записи каталога файлов. Максимальное возможное количество запоминаемых старых состояний зависит от размера блока файла. Если происходит переполнение нулевого блока, то самое старшее состояние теряется и об этом выдается соответствующее сообщение.

Литература

1. Изотамм А., Каазик Ю., Томбак М., Язык определения записи. Труды ВЦ ТГУ, 1978, № 41, 7-64.
2. Кнут Д., Исскуство программирования для ЭВМ. т. 3, "Мир", М., 1978.
3. Каазик Ю., Образование файлов. Труды ВЦ ТГУ, 1978, № 41, 65-74.
4. Каазик Ю., Томбак М., Совместное пользование данными в системе РАМА. Наст. выпуск.

ПРЕДСТАВЛЕНИЕ ЗАПИСИ В СИСТЕМЕ РАМА

А. Изотамм

Вопросы описания и представления записи рассматривались впервые в статьях [1] и [2]. За прошедшие два года в эти проекты по разным причинам приходилось внести некоторые изменения. В настоящей статье описываются как эти причины, так и введенные коррективы. При этом предполагается, что читатель не интересуется всеми подробностями структур данных.

1. Структура физической записи

В ходе реализации проекта РАМА оказалось, что физическое представление записи, описанное в [2], имеет заметные недостатки. Поэтому в систему внесены соответствующие коррективы, для описания которых сперва вкратце напомним старую версию структуры дерева данных (физической записи).

В старой версии физическая запись представлялась как супермассив, находящийся в одной связной области памяти. Вершинам дерева соответствовали кодослова трех типов: кодослово типа "a" содержало ссылку на атом длиной выше 7 байтов, на упакованные данные или на таблицу организации, кодослово типа "b" содержало значение атома длиной до 7 байтов, а кодо-

слово типа "с" ссылалось на вектор кодослов более низкого уровня. Все ссылки были относительные в отношении адреса записи. Положительными чертами такой версии являются простота (а тем самым и скорость) выделения новых кусков памяти, и возможность ввода-вывода записей без любых дополнительных действий над адресами. Однако, требование изображения записи в одной связной области является неприемлемым, так как в общем случае невозможно найти оценок объема записи (если легендой определена хоть одна повторяющаяся группа без зафиксированного количества повторений). Это значит, что практически нельзя избегать переполнения поля создаваемой или модифицируемой записи, а для уменьшения количества переполнений следовало бы выделить память с большим запасом. Преодоление переполнения путем дублирования возможно тогда, когда в оперативной памяти имеется свободный связной участок длиной, превышающей длину переполненной записи. Кроме того, ввиду гашений отдельных поддеревьев в записи возникает необходимость "сборки мусора", а этим изменяются адреса сдвинутых данных, что усложняет проведение циклов в записи.

В новой версии мы отказались от требования связной области записи, сохраняя представление дерева в форме супермассива. Распределение памяти при этом полностью осуществляется средствами операционной системы (регистровая форма макрокоманд GETMAIN и FREEMAIN). Во избежание чрезмерного раздробления памяти (а также для сокращения числа обращений к GETMAIN и FREEMAIN) физическая запись строится как блочная структура - каждый блок занимает одно связное поле в памяти. Пользуясь терминологией, введенной в [3], можно сказать, что

блок - это ветка группы без корня этой группы. Кроме того, отдельные блоки выделяются также для таблиц организации, атомов неопределенной длины и внешних структур. Кодослово типа "c" содержит теперь абсолютную ссылку на блок[1], а кодослово типа "a" содержит относительную ссылку на атом или упакованную группу, расположенных в одном блоке с кодословом.

В системе различаются логические и физические блоки. Части физического дерева данных называются физическими блоками, а соответствующие им отрывки дерева описания данных - логическими блоками. Последние нумеруются следующим образом. Корень дерева легенды находится в нулевом блоке. Остальные блоки получают номера в том порядке, как их проходят при движении "корень-вниз-направо".

Рассмотрим, например, легенду

```
LEG ШКОЛЫ KEY=ШКОЛА.TEXT
* 1 ШКОЛА PICT=50
* 1 КОЛИЧ CONST NAT
* 1 КЛАСС REP=КОЛИЧ
  * 2 НОМЕР PICT=3
  * 2 КОЛИЧ CONST MAX=40
  * 2 УЧЕНИК REP=КЛАСС.КОЛИЧ PICT=20
      SORT KEY=ФАМ,ИМЯ
    * 3 ИМЯ
    * 3 ФАМ
    * 3 ПОЛ SCOPE=[М,Д]
  * 2 ПРЕДМЕТ REP PICT=8
END
```

---

1 Исключением является корень простой (неповторяющейся) группы, если такая группа входит в ту же ветку, где находится ее корень: кодослово здесь типа "c", а ссылка относительная.

В интересах краткости и наглядности введем следующий формат для вершины дерева легенды

Имя или вид вершины		
ссылка вниз	номер блока	ссылка направо или вверх

Вид вершины организации обозначается через "o", а вершины промежуточного уровня - через "n". Дерево легенды с указанием номеров его логических блоков приведено на рис. 1, на рис. 2 представлена блочная структура соответствующей физической записи.

Как видно, блоки таблиц организации не нумерируются. Блоки 6 и 7 нуждаются, вероятно, в объяснении, так как на значение атома должно вообще-то ссылать (как было сказано выше) кодослово типа "a". В данном случае значение атома является экземпляром группы (а тем самым и веткой), что обусловливает выделение самостоятельного блока; типом ссылки на блок является "c".

Между физическими и логическими блоками имеет место следующее соответствие. Во-первых, в любой непустой записи имеются по меньшей мере физические блоки, соответствующие логическим блокам с номерами 0 и 1. Во-вторых, если i-тый ($i > 1$) логический блок является потомком блока, соответствующего повторяющейся группе, то ему может соответствовать более одного физического блока. В-третьих, в физической записи могут существовать физические блоки, которые не определены логическими блоками: таким образом изображаются таблицы организации.

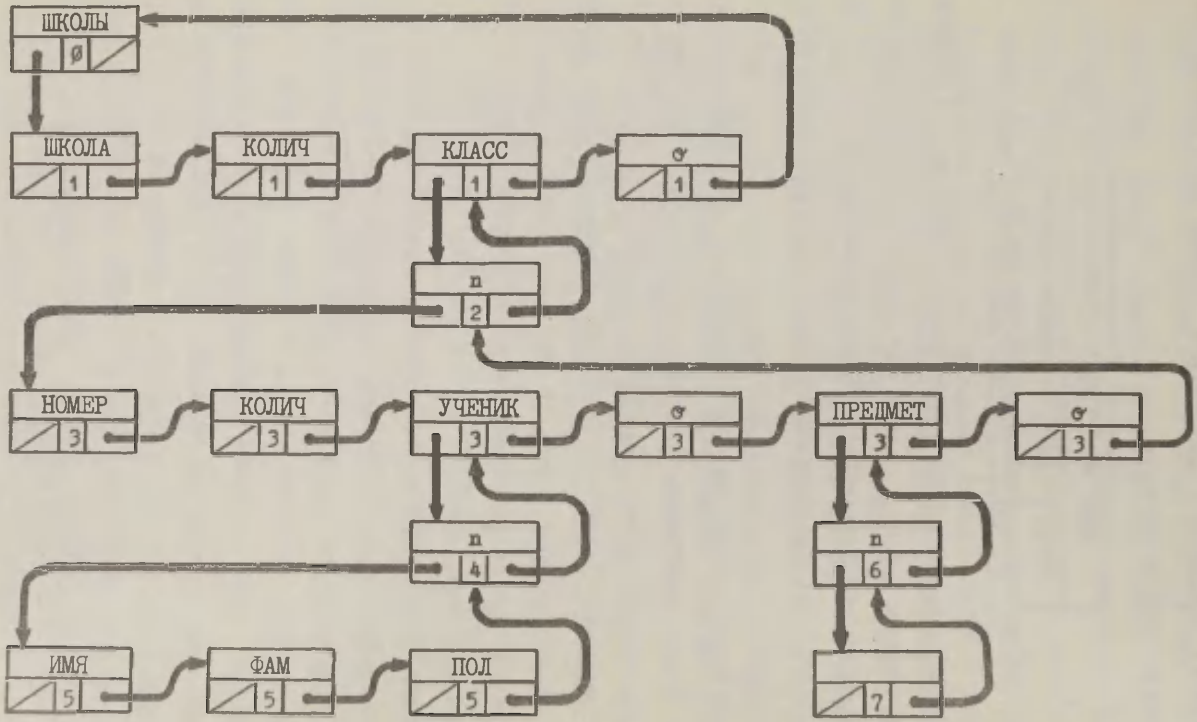

ШКОЛЫ
Ø
ШКОЛА
1
КОЛИЧ
1
КЛАСС
1
σ
1
n
2
НОМЕР
3
КОЛИЧ
3
УЧЕНИК
3
σ
3
ПРЕДМЕТ
3
σ
3
n
4
n
6
ИМЯ
5
ФАМ
5
ПОЛ
5
7

Рис. 1.

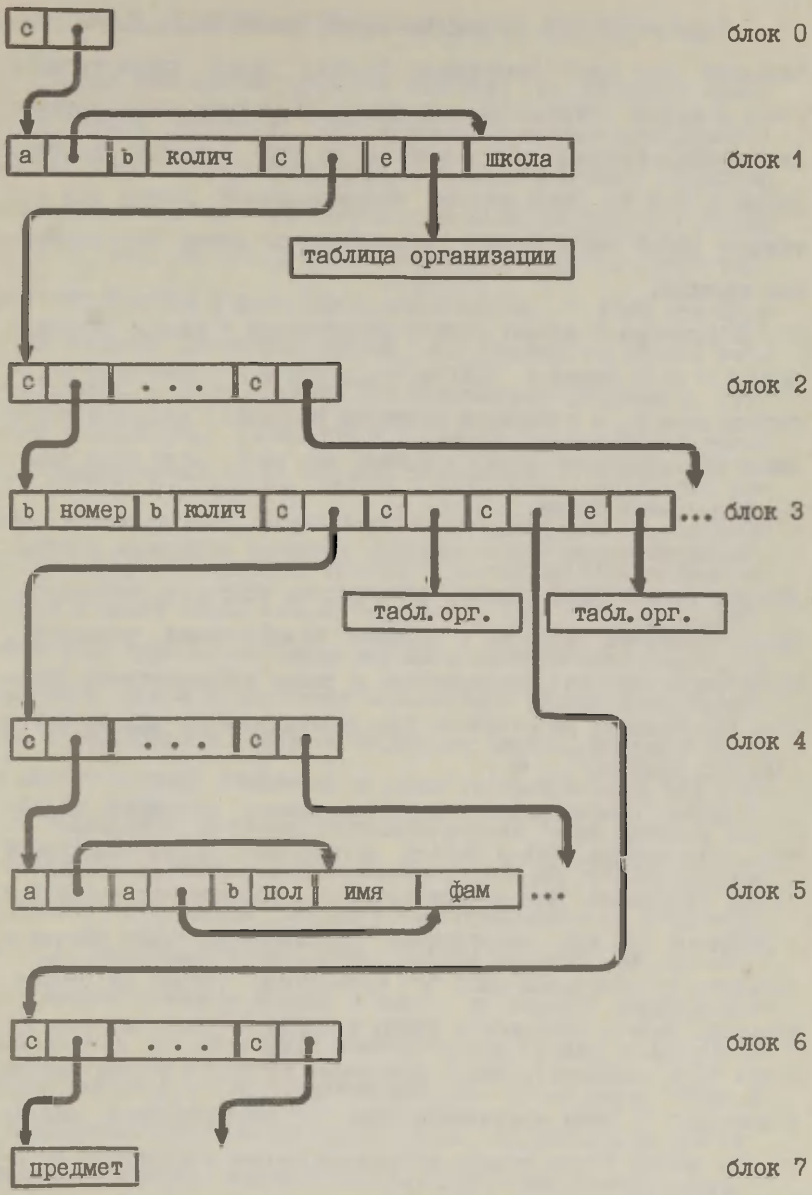

c
блок 0
a
b
колич
c
e
школа
блок 1
таблица организации
c
. . .
c
блок 2
b
номер
b
колич
c
c
c
e
... блок 3
табл. орг.
табл. орг.
c
. . .
c
блок 4
a
a
b
пол
имя
фам
...
блок 5
c
. . .
c
блок 6
предмет
блок 7

Рис. 2.

В зависимости от вычисления длины физического блока различаются два типа логических блоков: блоки фиксированной длины и блоки неопределенной длины. Последние соответствуют либо блокам промежуточного уровня (на рис. 2 такими являются блоки 2, 4 и 6), либо атомам неопределенной длины. Все остальные блоки имеют длины, вычисляемые во время транслирования легенды.

Информация о длинах блоков сохраняется в дереве описания записи в виде вектора BLSIZE(0:n), где n - количество логических блоков, а значением элемента BLSIZE(i) является длина блока фиксированной длины в байтах или нуль, если блок имеет неопределенную длину.

Преимуществами такого подхода являются экономное использование оперативной памяти, возможность удобного генерирования и обработки записей с заранее неизвестными размерами, устойчивые адреса поддеревьев, а также сравнительная простота операторов, необходимых при резервировании или же освобождении памяти.

Однако, рассмотренная структура записи усложняет операции ввода-вывода. Вывод записи происходит через системный буфер, программа заполнения которого проходит дерево записи в обратном порядке, освобождает переписанные блоки записи и заменяет в кодословах типа "c" абсолютные ссылки на относительные. Запись вводится с диска в связную область памяти, после чего соответствующая программа проходит дерево записи и заменяет во всех кодословах типа "c" относительные ссылки на абсолютные (если только кодослово вообще ссылает на физический блок).

2. Структура оттранслированной легенды

Опытная реализация процедур создания и обработки физической записи показала, что версия дерева описания записи, представленная в [1], является не совсем удачной. Основные недостатки являются следующими: во-первых, маркировка вершин (см. [1], стр. 10-15) не дает возможности для быстрого разветвления передач управления в зависимости от вида вершин и других свойств. Во-вторых, длиной поля вершины по старой версии являются 4 слова, причем дополнительная информация, связанная с вершиной, указывается ссылками на соответствующие поля: лишние переходы по ссылкам тратят время. В-третьих, создание индексируемых таблиц ключей, псевдоатомов и запасов значений не оправдывалось: доступ к элементам таблиц был неудобным и длины таких таблиц ограничены (см. [1], стр. 26-34). В-четвертых, переход на новую версию распределения памяти для физической записи предоставил возможность существенно упростить, а тем самым и ускорить обработку записи, перенеся многие работы с этапа обработки на этап транслирования легенды.

В результате в дерево описания данных были внесены существенные изменения. Можно сказать, что из старой версии сохранились в основном только общие принципы (см. [1], стр. 4-10) со следующими изменениями: каждая вершина дерева описания данных занимает теперь 8 слов, и вершина организации прибавляется любому корню повторяющейся группы, несмотря на наличие или отсутствие варианта доступа. Кроме того, если в повторяющейся группе встречаются атомы со свойством INDEX, то корню группы для каждого такого атома прибавляется дополнительная вершина организации.

Новый формат вершины представлен на рис. 3. Роль маркера выполняют теперь поля TOP, T и TYPE. Значением TOP является код вида вершины (предусматриваются 13 различных видов: корень дерева, корень массива, корень повторяющейся группы с организацией REP, то же с организацией LIST, три типа для вершин промежуточного уровня, корень альтернативной группы, корень простой группы, вершина организации, атом, сечение и внешняя структура). Значение поля TYPE указывает на тип кодослова в записи, который соответствует данной вершине. Обе названные поля закодируются так, что их значениями можно пользоваться для индексирования переходов в обрабатывающих программах. Информация поля T зависит от вида вершины. Для атома или сечения каждый двоичный разряд этого поля сигнализирует о наличии или отсутствии определенных свойств (константа, способ определения длины и т.п.), для корней повторяющихся групп, а также для корня альтернативной группы и вершины промежуточного уровня в массиве значение T указывает способ определения количества повторений.

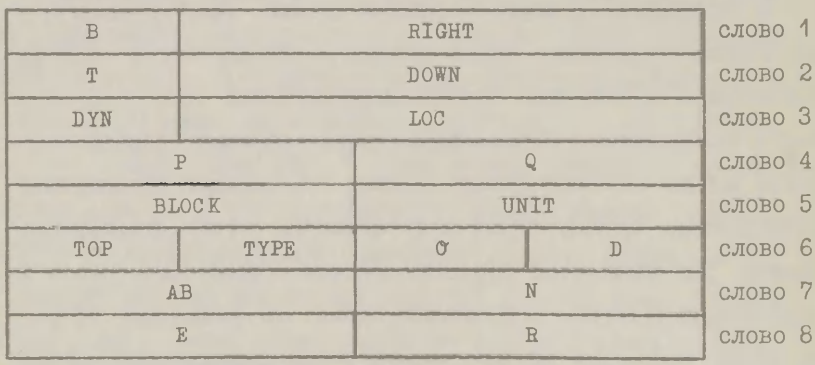


Рис. 3.

Поля B, RIGHT и DOWN предназначены для ссылок дерева так, как принято всюду в проекте PAMA: RIGHT ссылает либо на соседнюю вершину (тогда B = Ø), либо на отца, если данная вершина не имеет соседа (B = 1), DOWN ссылает либо на старшего сына, либо имеет пустое значение.

Подполе BLOCK содержит номер того блока, где в физической записи находится соответствующее кодослово (если данные не упакованы) или значение атома (если данные упакованы). Слова 3 и 4 в вершине задают макет кодослова, который состоит из подполей DYN, LOC, P и Q. Значением поля LOC является либо относительный адрес самого кодослова в своем блоке, либо (в случае упакованности данных) относительный адрес значения атома или сечения в блоке. Содержание подполя DYN задает короткую характеристику данных, связанных с этим кодословом: состояние каждого двоичного разряда имеет определенную семантику. Поля P и Q дают в зависимости от вида вершины информацию о длине ссылаемого блока (или атома).

Значением поля C является в случае атома его тип, в случае вершины организации - организация группы (для корня REP-группы - вариант доступа, для корня массива - количество индексов), а для вершины промежуточного уровня массива - порядковый номер соответствующего индекса массива. Подполем D пользуются лишь в том случае, если видом вершины является либо вершина организации (тогда значением D является код варианта доступа), либо атом (тогда D указывает на команду загрузки значения атома в регистр или пересылки этого значения на поле выводных данных, определенного пользователем). Значением поля AB в корнях является номер подчиненного логичес-

кого блока, а в атоме - относительный адрес значения атома в блоке (или номер блока значения атома, если он имеет неопределенную длину).

Подполе N предназначено в корне повторяющейся группы для ссылки на вершину того атома, значением которого задается количество повторений; в корне альтернативной группы ссылаемая вершина служит ключом выбора.

В новой версии вместо таблиц ключей, псевдоатомов и запасов значений создаются соответствующие цепочки, звенья которых заменяют записи таблиц старой версии. Поле N служит для размещения ссылки на звено цепи ключей в вершине организации и в корне дерева легенды (если запись снабжена ключом). Если атом (или сечение) имеет запас значений или свойство PSEUDO, то N ссылается на звено соответствующей цепи; если данные свойства имеются одновременно, то значением поля N является ссылка на слово, где первое полуслово ссылает на звено списка SCOPE, а второе полуслово - на звено PSEUDO.

Подполем E пользуются тогда, когда вершиной описывается либо корень повторяющейся группы (тогда значением E является номер логического блока экземпляра группы), либо корень простой группы (E = количество всех элементов в группе), либо атом, у которого имеется определенное в легенде максимальное значение (тогда E содержит ссылку на это значение), либо корень альтернативной группы (E = номер логического блока, содержащего ключ выбора).

Семантика значения поля R зависит также от вида вершины. В корне альтернативной группы, а также повторяющейся группы, если количество повторений определено значением атома, R ссы-

лает на относительный адрес этого атома или ключа выбора. В корне простой группы или в вершине промежуточного уровня значением поля R является количество вводимых элементов[2] группы. В случае атома с определенной длиной содержанием этого поля будет заданная в легенде длина – максимально возможное количество символов печатного изображения значения атома.

Содержание понятия "атом с неопределенной длиной" изменился в сравнении с предыдущими публикациями о проекте РАМА. Раньше таковым считали атом типа HEX или TEXT, длина которого не была явно задана в легенде. В новой версии в таком случае длиной атома по умолчанию устанавливается 8 байтов, а конструкция "длина" в языке RDL определяется так, как показано на рис. 4.

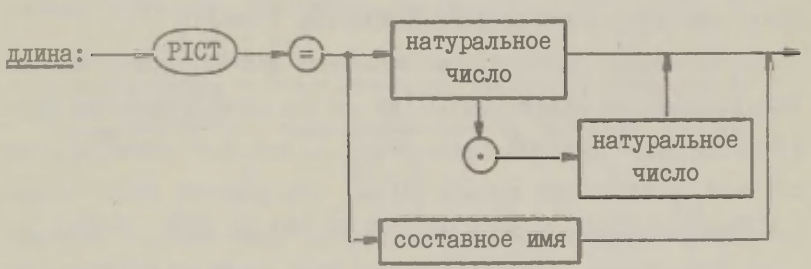


Рис. 4.

Составным именем ссылаются на доступный атом со свойством CONST, значение которого задает "неопределенную длину" данного атома. В таком случае поле R содержит ссылку на звено таблицы имен, соответствующее составному имени.

---

2 Не вводятся, например, псевдоатомы, таблицы организаций.

В случае сечения полю R присваивается значение некоторой функции от типа атома и заданных позиций сечения, что дает программам обработки записи возможность для быстрого выделения значения сечения.

Наконец, если вершина описывает корень внешней структуры, то единственной характеристикой является имя этой структуры, которое размещается в последнем двойном слове вершины. Поля T, TYPE, C и D, а также слова 2 и 3 не используются.

В новой версии изменилась также структура поля всей оттранслированной легенды. Это поле занимает и теперь одну связную область памяти, но состоит из заголовка, таблицы имен и области, где вперемешку следуют вершины дерева и звенья цепей псевдоатомов, запасов значений и ключей, а также звенья цепи омонимов, принадлежащие к таблице имен. В конце той области находится вектор длин логических блоков.

В заголовке представлены следующие данные: ссылки на корень дерева, на таблицу имен, на вектор длин логических блоков и на цепи ключей, псевдоатомов и запасов значений, имя легенды, а также имя набора данных, содержащего текст легенды и длина оттранслированной легенды. Кроме того, в заголовке резервированы поля для размещения системных ссылок во время работы с легендой. Общий вид цепей ключей, SCOPE и PSEUDO в оттранслированной легенде представлен на рис. 5.

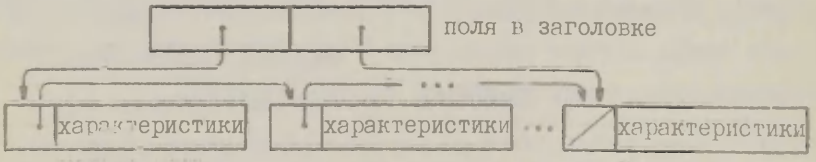


Рис. 5.

Каждой вершине организации соответствует по одному звену цепи ключей. Если запись имеет ключ(и), то ей также соответствует одно звено данной цепи, но таблица организации для нее не строится. В дальнейшем будем говорить, что звеном цепи ключей представляется замок, имеющий определенное в легенде количество ключей. В поле характеристик замка содержатся данные о замке и о всех ключах этого замка. К первым относятся, например, сведения о варианте доступа, организации повторяющейся группы, номера логических блоков корня, вершины промежуточного уровня и экземпляра повторяющейся группы, относительный адрес ссылки на таблицу организации в блоке корня, а также ссылки на соответствующие вершины дерева описания записи. Далее следуют характеристики отдельных ключей замка, каждый из которых занимает одно двойное слово памяти и имеет структуру, показанную на рис. 6.

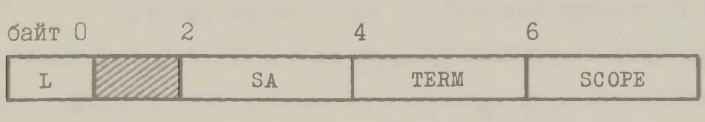


Рис. 6.

Подполе TERM ссылает на вершину дерева описания данных, соответствующую атому (или сечению) ключа, поле SA содержит относительный адрес значения ключа в блоке экземпляра, поле L предназначено для длины ключа в байтах, а поле SCOPE ссылает на звено цепи запасов значений, если атом ключа имеет такое свойство (иначе данное поле является пустым).

Если ключ задан в легенде явно (конструкциями KEY = ... или BASE(DOWN) = ...), то поле ключа используется естественным образом. В случае же массива количеством ключей счи-

тается число индексов; каждому из них соответствует также одно поле ключа, но непустое значение имеет такое поле лишь в том случае, когда индекс меняется по запасу значений ссылаемого в легенде атома. Тогда значением поля TERM является ссылка на ссылаемую составным именем вершину атома в дереве описания данных, а поле SCOPE ссылает на запас значений данного атома (ср. [3], стр. 33-34).

Характеристики цепи SCOPE остались в сравнении со старой версией без изменения ([1], стр. 31-34). Формат звена цепи PSEUDO остался также прежним ([1], стр. 28-30), но характеристики атома типа INDEX с дополнительным упорядочением транслируются по-новому: в зависимости от направления упорядочения введены два нового типа PSEUDO. Тип 16 соответствует направлению, заданному через BASE, а тип 20 - BASEDOWN. Теперь "ссылка P" ссылает на соответствующее звено цепи ключей (а не на таблицу параметров, как в старой версии).

Принципы построения таблицы имен и техника распознавания составных имен представлены в [1] и [4], там же описан принципиальный формат звена цепи омонимов. В настоящей версии проекта названное звено содержит многие дополнительные данные, такие, как номер логического блока и относительный адрес кодослова в этом блоке, ссылка на вектор ссылок на замки, которые находятся на пути с корня дерева до описываемой вершины. Для атома (или сечения) прибавляются ссылки на звенья списков псевдоатомов и запасов значений (если описываемый атом имеет такие свойства), а также ссылка на вектор ссылок на звенья цепи замков, где данный атом (сечение) служит ключом.

Литература

1. Изотамм А., Дерево описания записи в системе РАМА. Труды ВЦ ТГУ, 1980, № 43, 3-35.
2. Изотамм А., Физическое представление записи в системе РАМА. Труды ВЦ ТГУ, 1980, № 43, 36-54.
3. Изотамм А., Каазик Ю., Томбак М., Язык определения записи. Труды ВЦ ТГУ, 1978, № 41, 7-64.
4. Изотамм А., Техника распознавания составных имен. Труды ВЦ ТГУ, 1980, № 45, 3-20.

ЯЗЫК ВЫВОДА ДАННЫХ

Я. Пейал, В. Соо, М. Томбак

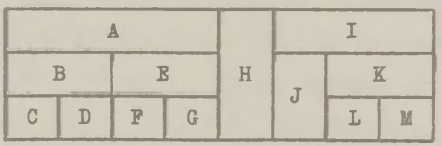
В статье кратко описывается язык DOL (= Data Output Language) для вывода данных в виде таблиц в системе РАМА. Язык DOL является развитием идеи, на которых основывается язык оформления таблиц в системе VILLIS (см. [6]).

Для чтения настоящей статьи требуется знакомство с системой РАМА в объеме статьей [1-5], [7].

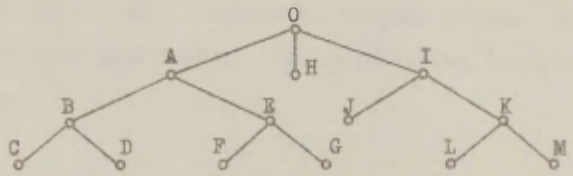
1. Неформальное введение

Данные в системе РАМА имеют древовидную структуру. Такую же структуру имеет заголовок ("шапка") таблицы (хотя иногда надо добавить фиктивный корень).

Пример 1. Пусть шапка таблицы имеет следующий вид:



тогда этой шапке соответствует следующее дерево



Вершину 0 (фиктивный корень) здесь нам пришлось добавить для того, чтобы превратить множество деревьев в дерево.

Язык DOL основывается на гипотезе, что такое сходство структур данных системы PAMA и таблиц не случайно, а отражает сущность явлений. Следовательно, таблицы можно всегда сопоставить с данными (а внешний вид каждой таблицы с некоторым описанием записи, т.е. легендой). Язык DOL представляет собой язык для задавания таких соответствий.

Пример 2. По дереву из примера 1 можно построить легенду

```
LEG 0 NAT PICT=3
  * 1  A
    * 2  B
      * 3  C
      * 3  D
    * 2  E
      * 3  F
      * 3  G
  * 1  H
  * 1  I
    * 2  J
    * 2  K
      * 3  L
      * 3  M
END
```

Любую запись файла с этой легендой можно представить в виде однострочной таблицы с шапкой из примера 1. Если в некоторой записи атомы уже имеют конкретные значения, то такую запись естественно представить в виде таблицы:

A				H	I		
B		E			J	K	
C	D	F	G			L	M
3	347	15	770	0	15	4	548

Прямоугольники из шапки таблицы, соответствующие вершинам дерева, будем называть "ортогснами". Наиболее естественно печатать в каждый ортогон имя соответствующей вершины (или имя оформления). Так мы и поступаем, но предоставляем пользователю и возможность заполнить ортогон другим текстом.

В примере 2 мы имеем взаимно-однозначное соответствие между вершинами дерева легенды и ортогонами заголовка таблицы. Если легенда содержит массивы и повторяющиеся группы, то такого соответствия уже не будет.

Пример 3. Рассмотрим легенду

```
LEG ПРИМ DEC PICT=3
  * 1  H
    * 2  A
    * 2  B
      * 3  P  ARRAY[3]
      * 3  Q  ARRAY[4]
  * 1  T
END
```

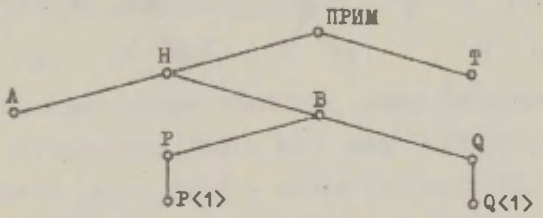
и соответствующую таблицу

H								
A	B							T
	P			Q				
	1	2	3	1	2	3	4	
a	p_1	p_2	p_3	q_1	q_2	q_3	q_4	t

В этом примере мы имеем взаимно-однозначное соответствие со структурой записи, а не легендой. Структура записи наиболее удобно представляется деревом описания записи (ср. [1]). Это дерево легенды, в которое добавлены т.н. "промежуточные" вершины, соответствующие ключам или же индексам массивов.

В DOL-программе надо ссылаться на ортогоны шапки таблицы (хотя бы для того, чтобы указать текст, который следует печатать в данный ортогон). В качестве таких ссылок используем имена вершин дерева описания записи. Для ссылок на простые вершины служат их составные имена (см. [1,2]), а на промежуточную вершину — составное имя соответствующей повторяющейся группы вместе с порядковым номером уровня в скобках < >.

Пример 4. Дерево описания записи для легенды ПРИМ примера 3 выглядит так (у вершин указаны их минимальные имена):



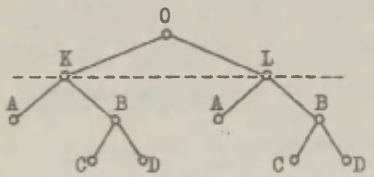
Если сравнивать дерево примера 4 с шапкой таблицы примера 3, то увидим, что трем ортогонам, которые находятся под ортогоном P, соответствует одинаковое имя P<1>, а четырем соседним ортогонам имя Q<1>. Такая множественность объектов с одним именем вынуждает нас дать этим ортогонам одинаковые предписания оформления. Это естественно, так как основное применение понятие повторяющейся группы находит при неизвестном или неопределенном количестве элементов группы.

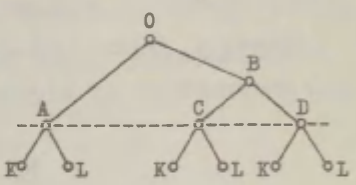
Иногда надо печатать таблицы, содержащие более одной строки, причем шапка строк тоже имеет свою структуру. Оказывается, что и с такими таблицами можно сопоставить легенды (специального вида). Возможны две разные трактовки таблиц: или шапка столбцов определяет структуру каждой строки, или же шапка строк определяет структуру каждого столбца.

Пример 5. Таблице вида

	A	B	
		C	D
K			
L			

можно ставить в соответствие следующие два дерева:





Прерывистая линия разбивает каждое дерево на две части. Все, что остается ниже этой линии, составляет в левом дереве шапку столбцов, а в правом - шапку строк. Отметим, что поддеревья, корнями которых являются вершины на прерывистой линии, имеют как в левом, так и правом дереве одинаковый вид, что, конечно, так и должно быть.

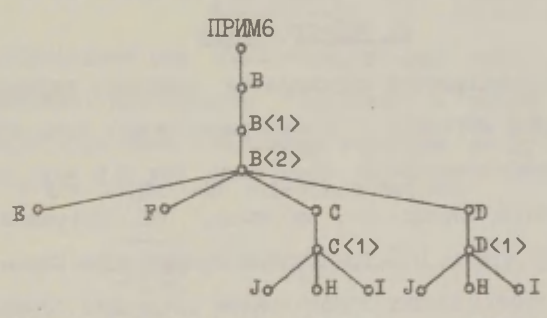
В языке DOL допускается отклонение от отмеченного правила: можно отсекать некоторые лишние ветви дерева и только после этого все поддеревья, начинающиеся с разбиения строк и столбцов, должны иметь одинаковый вид. Список листьев дерева, который получается после удаления лишних ветвей, назовем нижним разбиением. Список же вершин, разделяющих шапки строк и столбцов, назовем средним разбиением.

Чтобы иметь возможность печатать по одной DOL-программе множество таблиц (т.е. одну трехмерную таблицу), вводим еще верхнее разбиение: все поддеревья его вершин после удаления лишних частей опять должны иметь одинаковый вид.

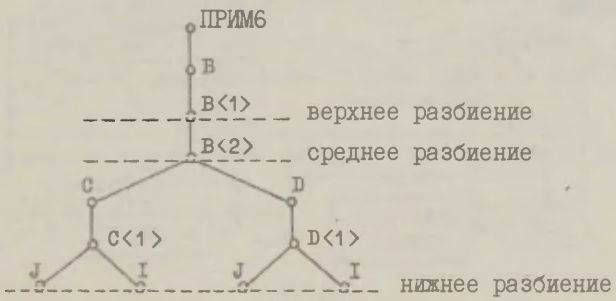
Пример 6. Легенде

```
LEG ПРИМ6 TEXT PICT=2
  * 1  B  REP  KEY=E,F
    * 2  E
    * 2  F
    * 2  C  REP  KEY=H
      * 3  J
      * 3  H
      * 3  I
    * 2  D  LIKE=C
END
```

соответствует дерево описания записи



Отсекаем все листья, которые используются в качестве ключей. Тогда нижнее разбиение состоит из четырех вершин C.J, C.I, D.J, D.I. Пусть среднее разбиение задано вершиной B<2> (поддеревья ниже этого разбиения определяют шапку строк), а верхнее разбиение - вершиной B<1>. Тогда получаем дерево:



Перечисляем теперь коротко возможности языка DOL. Этот язык позволяет:

1) определить критерии для выбора данных;
2) дать разбиение дерева на таблицы, строки и столбцы;
3) дать предписания для оформления комплекта таблиц;
4) запретить печатать ненужные ортогоны;
5) соединять ортогоны промежуточных уровней в один ортогон;
6) управлять оформлением ортогонов;
7) управлять оформлением данных (содержанием таблицы).

2. DOL-программа

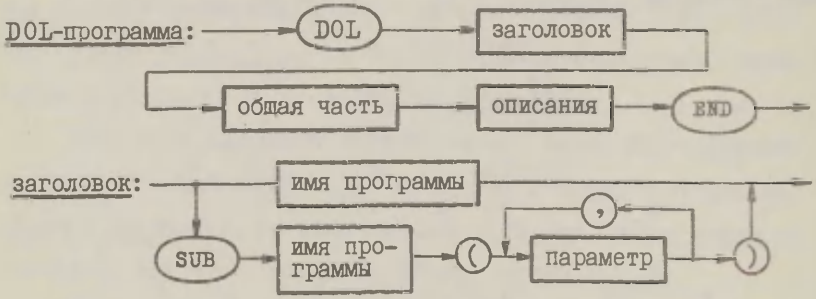
Одной DOL-программой описывается комплект таблиц, печатаемых по одной легенде. DOL-программа может быть оформлена как в виде самостоятельной программы, так и в виде подпрограммы (к которой обращается, например, DML-программа).

Синтаксис языка DOL представим в виде схем Вирта, в которых кроме традиционных использованы следующие обозначения:

[c] - целое без знака;

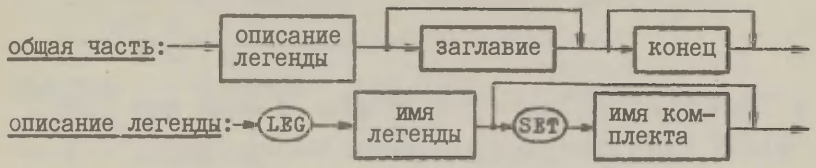
[s] - строка длиной не более 255 символов, заключенная между апострофами;

[I] - идентификатор.

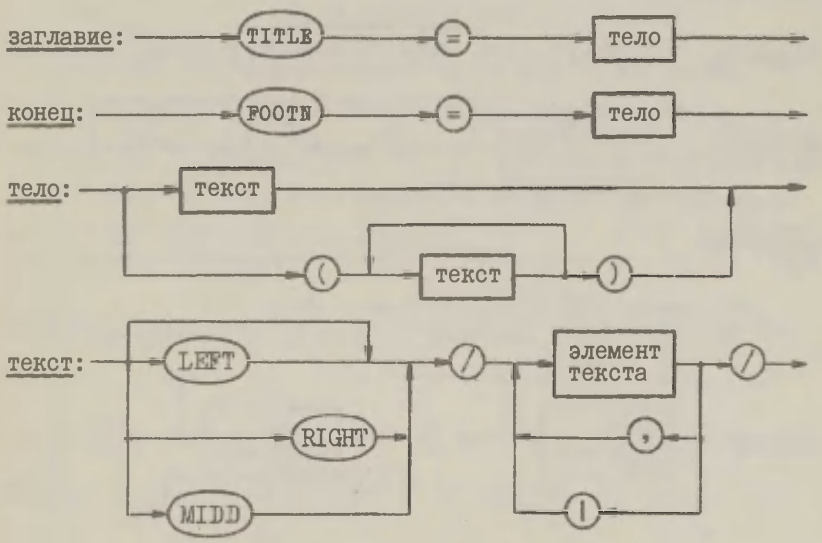


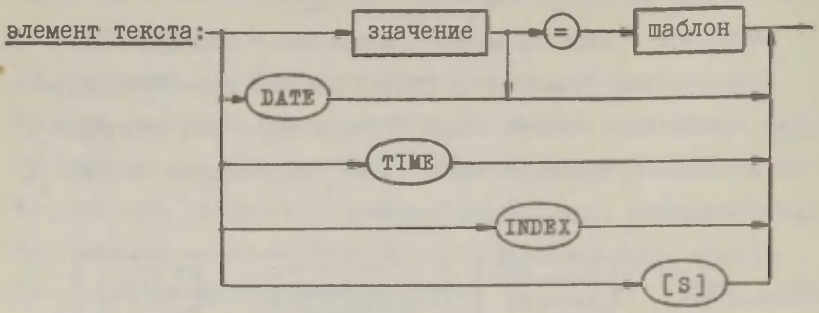
Параметры - идентификаторы, локальные в DOL-программе.

В приводимых схемах всем синтаксическим конструкциям, которые начинаются словом "имя" (например "имя программы", "имя легенды" и т.д.), соответствует терминальный символ [I]. Соответствующие схемы будем опускать.

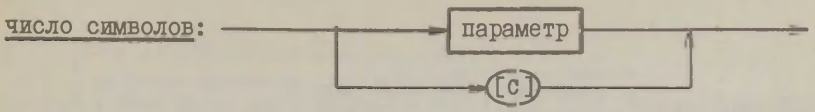


Если отсутствует имя комплекта, то оно отождествляется с именем легенды. Конструкции "заглавие" и "конец" в общей части относятся ко всем печатаемым таблицам, но их можно использовать и для оформления отдельных таблиц.





Элемент текста может быть печатан в левый край (LEFT), в правый край (RIGHT) или в середину таблицы (MIDD). По умолчанию текст печатается в середине таблицы. Элементом текста могут быть строки (разделитель | означает переход на новую строку), значения атомов (при надобности вместе с шаблоном для печати), текущее число (DATE), время (TIME) и номер таблицы (INDEX).

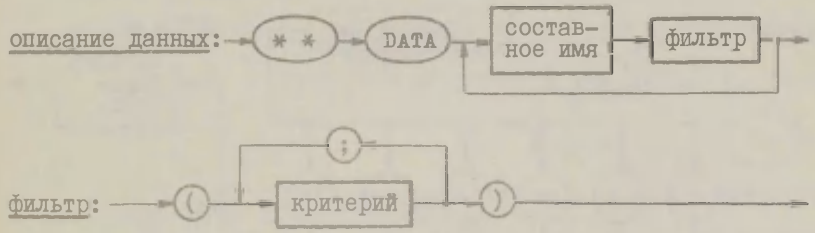


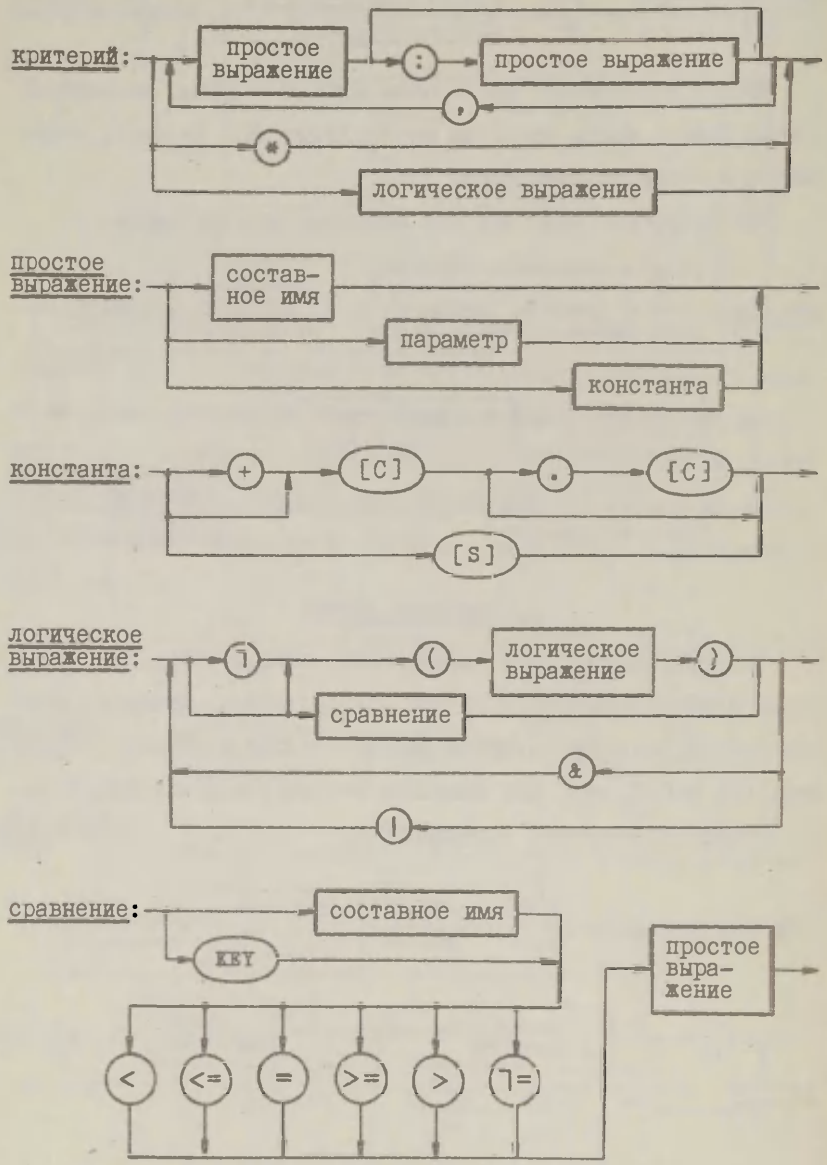
Шаблон определяет число символов до и после десятичной точки. Если в конце шаблона задана буква E, то число печатается в полулогарифмическом виде.

DOL-программа содержит еще описания четырех типов.

3. Описание данных

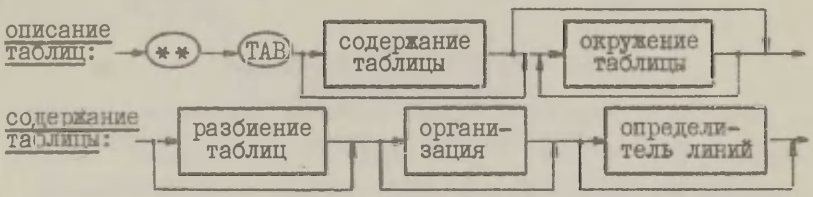
Описание данных дает предписания выбора тех данных, которые следует печатать в данном наборе таблиц. Описание данных состоит из списка вершин вместе с фильтрами (см. [4], стр. 119 и [7], стр. 52; символом * обозначается тождественно истинное логическое выражение).



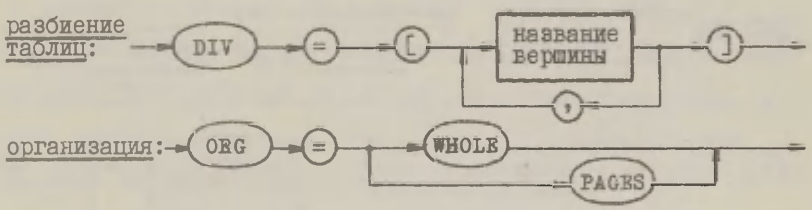

критерий:
простое выражение
:
простое выражение
,
*
логическое выражение
простое выражение:
состав-ное имя
параметр
константа
константа:
+
[C]
.
[C]
[S]
логическое выражение:
¬
(
логическое выражение
)
сравнение
&
|
сравнение:
составное имя
KEY
простое выра-жение
<
<=
=
>=
>
¬=

4. Описание таблиц

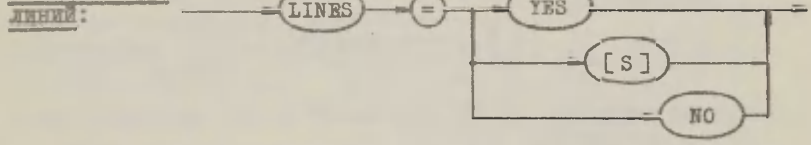
Описанием таблиц задается разбиение таблиц, организация печати (по страницам или целыми таблицами), заглавия и концы таблиц, а также расположение линий в таблице.



Разбиение таблиц дает список вершин, каждый из которых определяет одну таблицу (т.е. верхнее разбиение). По умолчанию DOL-программа выдает одну таблицу на запись.

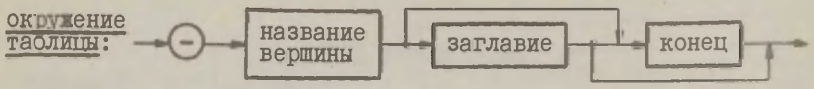


В случае ORG = PAGES таблицы оформляются в виде 60-строчных страниц, оставляя между страницами 12 пустых строк. В случае ORG = WHOLE (и по умолчанию) печатаются целые таблицы.



При LINES = YES или LINES = [S] (и по умолчанию) в таблице печатаются все линии (если при помощи строки не указано иначе, то вертикальные линии печатаются символом "|" и горизонтальные символом "-"). В описаниях строк и столбцов мож-

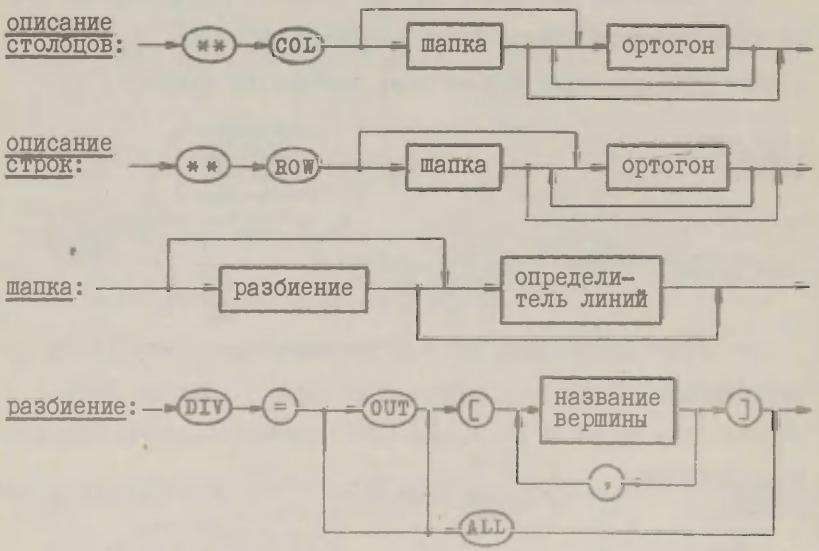
но дополнительно уточнять вид печати линии.



Эта конструкция позволяет оформлять окружение отдельных таблиц. Название вершины используется здесь в качестве имени таблицы и поэтому должно входить в разбиение таблиц.

5. Описания строк и столбцов

Описанием столбцов (строк) задаются разбиение столбцов (строк), вид печати вертикальных (горизонтальных) линий и ортогоны заголовка (левого края) таблицы.



Порядок описания строк и столбцов в DOL-программе имеет существенное значение: впереди должно быть то описание, которое содержит среднее разбиение.

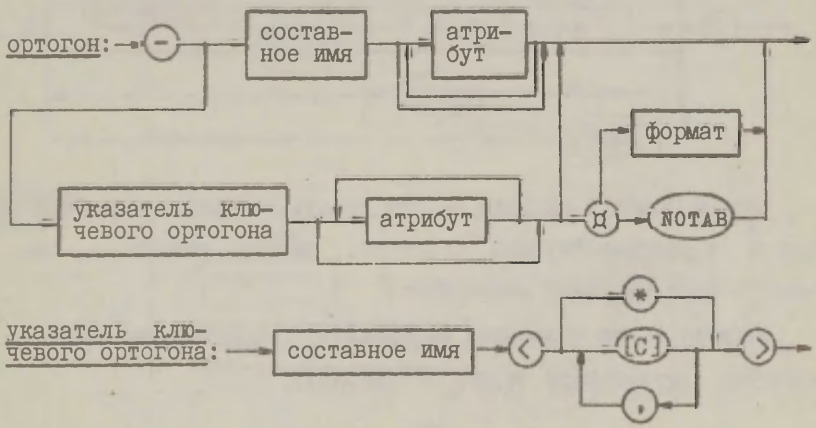
Варианты DIV = ALL и DIV = OUT[...] допускаются только при задавании нижнего разбиения. DIV = ALL имеет место по умолчанию и равносильно полному перечню всех листьев поддеревьев, начинающихся с вершин среднего разбиения. DIV = OUT указывает, что указанные в скобках вершины не будут включены в нижнее разбиение.

6. Описание ортогона

Все ортогоны можно делить на следующие два класса.

Основные ортогоны, соответствующие вершинам дерева легенды, т.е. таким вершинам дерева описания записи, которые имеют составное имя.

Ключевые ортогоны, соответствующие промежуточным вершинам дерева описания записи. Ключевые ортогоны состоят в общем случае из двух частей: часть имени, где печатается имя индекса (компоненты ключа), и часть значения. Если индекс не имеет имени, то эта часть отсутствует. В описании ортогона указанные части разделяются символом "Д".



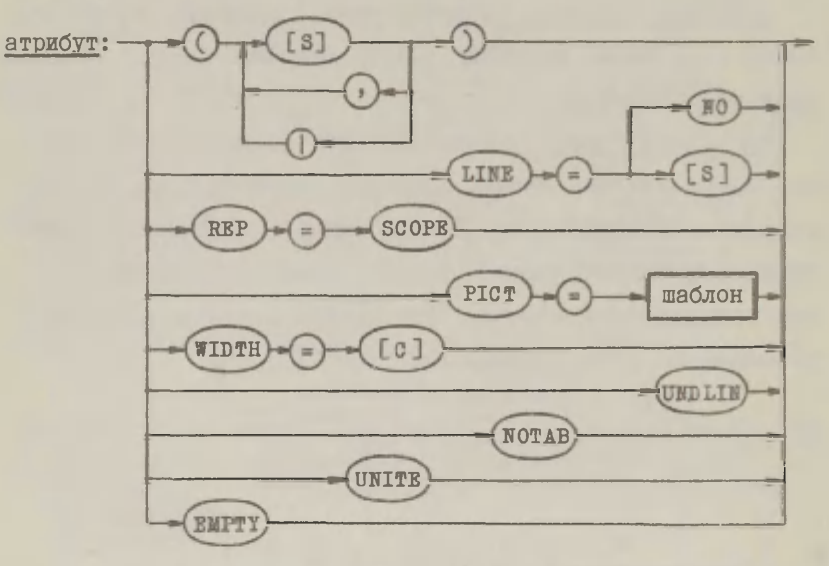
Запись <*> в указателе означает, что объединяются все ключи повторяющейся группы или все индексы массива.

Пример 7. Если легенда содержит фрагмент

```
* 4  A REP KEY=I,J,K
```

то запись A<*> равносильна записи A<1,2,3>. В части имени такого составного ортогона печатаются имена ключей, разделенные запятыми, а в части значения действительные значения (в том же порядке).

Основными средствами задавания вида таблицы являются атрибуты:



Список строк в скобках задает текст, который следует печатать в соответствующем ортогоне. UNDLIN указывает, что текст в ортогоне надо подчеркнуть.

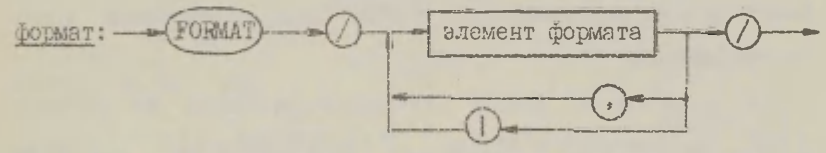
Атрибут WIDTH позволяет определить "собственную" ширину ортогона (минимальную ширину в символах).

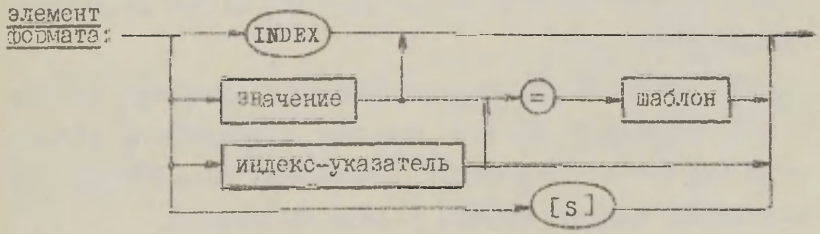
Атрибут LINE позволяет управлять печатью линий непосредственно влево (в случае столбцов) или сверху (в случае строк) от ортогона. Атрибут NOTAB запрещает соответствующий ортогон. В случае ключевых ортогонов надо запретить обе части отдельно.

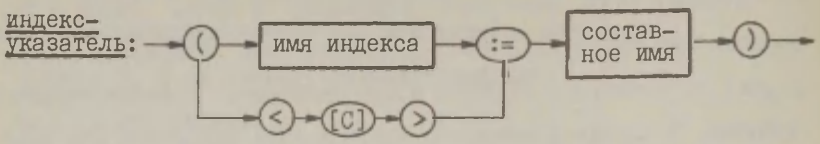
Атрибут UNITE имеет смысл только в вершинах, для которых в легенде определено сечение. С атрибутом UNITE такая вершина рассматривается как неделенный атом (если вершина имеет тип DATE или FDATE, то ее части разделяются точками).

Атрибутом REP = SCOPE можно воспользоваться в вершинах, соответствующих промежуточным уровням повторяющихся групп (массивов) с ключом (индексом) типа SCOPE. Использование указанного атрибута требует создания ортогонов для всего запаса значений, независимо от конкретных данных.

Атрибуты PICT и EMPTY управляют печатью данных в таблице: атрибут PICT дает шаблон, а атрибут EMPTY запрещает печатать данные, являющиеся потомками этой вершины (так можно оставить в таблице пустые строки и столбцы).







Часть значений ключевого ортогона при помощи формата можно заполнить ключом или индексом (конструкция "значение"), порядковым номером элемента повторяющейся группы (INDEX) или элементом некоторого другого массива, выбранного по индексу текущего элемента данной повторяющейся группы (конструкция "индекс-указатель"). Индекс-указатель работает следующим образом: вместо значения индекса, указанного слева от ":=" печатается значение соответствующего элемента одномерного массива (повторяющейся группы), указанного составным именем в правой части присваивания.

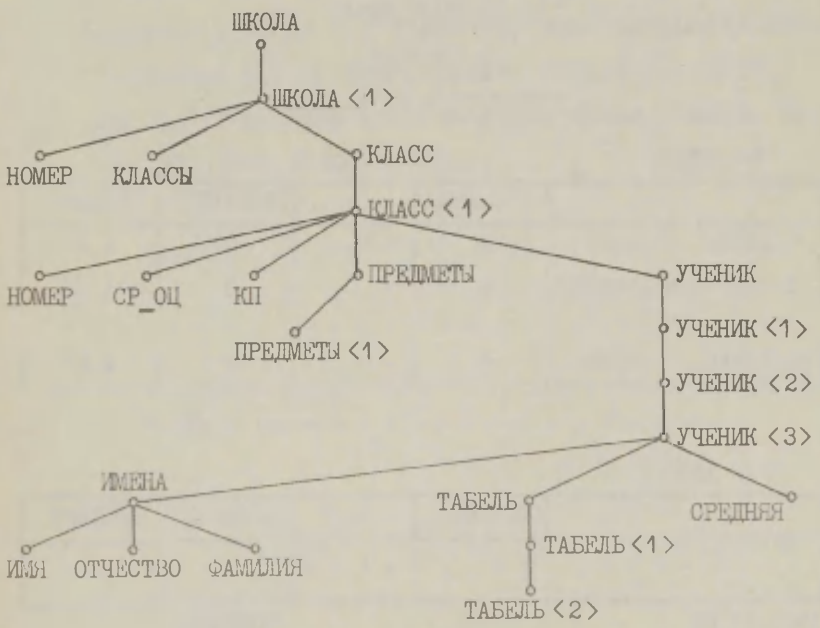
7. Пример

Приведем теперь сравнительно объемистый пример, который демонстрирует использование большинства конструкций языка вывода данных.

На странице 59 сверху изображен фрагмент из легенды ШКОЛА, приведенной в статье [3] (на странице 54). Соответствующее дерево описания записи приведено на той же странице снизу.

На странице 60 сверху приведена DOL-программа, которая по рассматриваемой легенде печатает совокупность таблиц, общий вид которых изображен на странице 60 снизу. Данные для составления этих таблиц нами выбраны произвольно.

```
LEG  ШКОЛА KEY = НОМЕР TEXT
  *1 НОМЕР DEC MAX = 1ØØ
  *1 КЛАССЫ CONST  'К-ВО КОМПЛЕКТОВ'
  *1 КЛАСС REP = КЛАССЫ  KEY = НОМЕР  SORT
    *2 НОМЕР PICT = 5
    *2 СР_ОЦ REAL  PSEUDO  PICT = 1.2
    *2 КП CONST MAX = 2Ø  'К-ВО ПРЕДМЕТОВ'
    *2 ПРЕДМЕТЫ ARRAY[КП] PICT = 8
    *2 УЧЕНИК KEY = ФАМИЛИЯ,ИМЯ,ОТЧЕСТВО SORT
      *3 ИМЕНА
        *4 ИМЯ
        *4 ОТЧЕСТВО
        *4 ФАМИЛИЯ
      *3 ТАБЕЛЬ ARRAY[4,КП]  NAT  MAX = 5  'ПЕРВЫЙ ИНДЕКС -
        СЕМЕСТР,ВТОРОЙ - ПРЕДМЕТ,ЭЛЕМЕНТ МАССИВА - ОЦЕНКА'
      *3 СРЕДНЯЯ REAL  PSEUDO  PICT = 1.2
END
```



```
DOL  ОЦЕНКИ  LEG  ШКОЛА
   TITLE=(#LDD/(НОМЕР)=2,'СРЕДНЯЯ ШКОЛА' !
     '2.СЕМЕСТР' ! 'УСПЕВАЕМОСТЬ ПО КЛАССАМ'/)
   FOOTN=(LEFT/DATE/   RIGHT/'ПОДПИСЬ: '/)
** DATA  ШКОЛА (НОМЕР=5)  ТАБЕЛЬ (2;*)
** TAB  DIV=[КЛАСС<1>]
   -КЛАСС<1>  TITLE=(LEFT/(КЛАСС.НОМЕР),'КЛАСС'/)
** ROW  DIV=[УЧЕНИК<3>]  LINES=NO
   -УЧЕНИК  NOTAB
   -УЧЕНИК<*>  NOTAB ¤ FORMAT/INDEX,(ФАМИЛИЯ)=8,',',(ИМЯ)=8/
** COL  DIV=[ТАБЕЛЬ<2>,СРЕДНЯЯ]
   -ТАБЕЛЬ  NOTAB
   -ТАБЕЛЬ<1>  NOTAB ¤ NOTAB
   -ТАБЕЛЬ<2>  NOTAB PICT=1 ¤ FORMAT/(<2>:=ПРЕДМЕТЫ)=8/
   -СРЕДНЯЯ  PICT=1.2
END
```

5.СРЕДНЯЯ ШКОЛА

2.СЕМЕСТР

УСПЕВАЕМОСТЬ ПО КЛАССАМ

1.А КЛАСС

	РУС.ЯЗЫК	...	ФИЗКУЛЬТ	СРЕДНЯЯ
1.ААМОС ,ТИЙУ	5	...	5	4.86
2.ААРЕСИЛЬД,ХЕЛГЕР	5		5	4.51
. . . . .				
38.ТИКАН ,ТАРМО	5		5	4.67

. . . . . .

11.Г КЛАСС

	МАТЕМАТ.	...	ФИЗКУЛЬТ	СРЕДНЯЯ
. . . . .				

1982.Ø2.Ø6 ПОДПИСЬ:

Литература

1. Изотамм А., Дерево описания записи в системе РАМА. Труды ВЦ ТГУ, 1980, № 43, 3-35.
2. Изотамм А., Техника распознавания составных имен. Труды ВЦ ТГУ, 1980, № 45, 3-20.
3. Изотамм А., Каазик Ю., Томбак М., Язык определения записи. Труды ВЦ ТГУ, 1978, № 41, 7-64.
4. Каазик Ю., Рауп А., Язык манипулирования данными. Труды ВЦ ТГУ, 1978, № 41, 97-140.
5. Каазик Ю., Томбак М., Система РАМА для управления базой данных. Труды ВЦ ТГУ, 1978, № 41, 3-6.
6. Маллинг А., Нигуль А., Томбак М., Язык оформления таблиц в системе VILLIS. Труды ВЦ ТГУ, 1977, № 39, 79-111.
7. Рауп А., Реализация языка манипулирования данными. Труды ВЦ ТГУ, 1980, № 45, 36-65.

РЕАЛИЗАЦИЯ ОДНОВИЗИТНЫХ АБСТРАКТНЫХ АТРИБУТНЫХ ГРАММАТИК

М. Мериете

Важной проблемой применения атрибутных грамматик в системах построения трансляторов является организация вычисления атрибутов в процессе трансляции. Нынешнее состояние в области атрибутных систем построения трансляторов характеризуется, прежде всего, нетехнологичностью этих систем и низкой эффективностью построенных трансляторов [9]. С целью преодоления отличенных трудностей в рамках технологического подхода [2] к автоматизации построения трансляторов разработан аппарат абстрактных атрибутных грамматик (ААГ) [1].

ААГ отличаются от классических атрибутных грамматик [3] наличием средств явного описания семантики итеративных языковых конструкций. Это достигается применением регуляризованных КС-грамматик (вместо КС-грамматик) для описания синтаксиса и введением специального вида семантических правил для вычисления атрибутов итеративных синтаксических структур.

В настоящей статье коротко описывается аппарат абстрактных атрибутных грамматик, рассматриваются алгоритм построения семантического вычислителя и алгоритм вычисления атрибутов для одного подкласса ААГ.

1. Абстрактные атрибутные грамматики

Пусть $G = (V_N, V_T, P, S_0)$ – регуляризованная КС-грамматика [8], где V_N и V_T обозначают множества нетерминальных и терминальных символов соответственно, P – множество правил вывода и S_0 – аксиому. Обозначим порождаемый грамматикой G язык через $L(G)$. Правило вывода с порядковым номером p имеет вид

$$p : Y \rightarrow \alpha ,$$

где $Y \in V_N$ и α – регулярное выражение в алфавите $V = V_N \cup V_T$. Обозначим i-тый слева символ в p-том правиле вывода через $p[i]$, $i = 0,\ldots,n_p$, где $n_p \geq 1$ – количество символов из V в правой части p-того правила вывода.

Построим абстрактную атрибутную грамматику AG, исходя из грамматики G, следующим образом. Каждым символом $X \in V$ сопоставляется множество атрибутов

$$A(X) = I(X) \cup S(X),$$

где $I(X)$ и $S(X)$ обозначают множества унаследованных и синтезированных атрибутов символа X. Предполагается, что AG сбалансирована, т.е. для любого $X \in V$

$$I(X) \cap S(X) = \emptyset .$$

Кроме того, для каждого $X \in V$ выделяются еще следующие подмножества атрибутов. Через

$$SEM(X) \subseteq A(X)$$

обозначается множество семантически существенных атрибутов символа X. $SEM(X)$ содержит атрибуты, значения которых определяют семантику языковой конструкции, соответствующей символу X и, таким образом, сохраняются в атрибутированном де-

реве. Атрибуты из $A(X) \setminus SEM(X)$ служат вспомогательными переменными при описании семантики.

$$EXT(X) \subseteq A(X)$$

обозначает множество внешних атрибутов, значения которых определяются не средствами AG, а внешней средой до вычисления атрибутов из $A(X) \setminus EXT(X)$.

Предполагается, что для каждого $X \in V$ множества $SEM(X)$ и $EXT(X)$ отдельно задаются, причем допускается пересечение этих множеств. В классических атрибутных грамматиках используются два варианта определения этих множеств:

в большинстве работ (напр. [3,6]) принимается

$$SEM(X) = \begin{cases} S(X), & X = S_0 \\ \emptyset, & \text{иначе} \end{cases};$$

в работе [7] для каждого $X \in V$ принято

$$SEM(X) = A(X) .$$

В первом случае нельзя отделить генерацию объектного кода от вычисления атрибутов, а во втором случае возникают трудности с распределением памяти для значений атрибутов.

Будем говорить, что правило вывода $p : Y \to \alpha$ имеет вхождение атрибута a в i-той позиции ($0 \le i \le n_p$), если $a \in A(p[i])$. Обозначим такое вхождение атрибута через $p[i].a$. Кроме того, если p-тое правило содержит в правой части итеративное подвыражение

$$(p[i] \ldots p[i+k])^+ ,$$

то в случае $1 \le j \le i+k$ вхождение $p[j].a$ называется итеративным. С правилом вывода p (т.е. с правилом с номером p) сопос-

тавлены следующие множества вхождений атрибутов:

$$A(p,i) = \{\, p[i].a \mid a \in A(p[i]) \},\ 0 \le i \le n_p \ ,$$

$$I(p,i) = \{\, p[i].a \mid a \in I(p[i]) \},\ 0 \le i \le n_p \ ,$$

$$S(p,i) = \{\, p[i].a \mid a \in S(p[i]) \},\ 0 \le i \le n_p \ ,$$

$$A(p) = \bigcup_{i=0}^{n_p} A(p,i) \ .$$

Таким образом, атрибуты сопоставлены с символами алфавита грамматики G, а вхождения атрибутов – с правилами вывода этой же грамматики.

С каждым правилом вывода p из P сопоставляются семантические правила вычисления значений элементов из A(p): с каждым $p[0].a \in S(p,0)$ сопоставляется правило $f^p_{a,0}$, а с каждым $p[i].a \in I(p,i)$ при $1 \le i \le n_p$ правило $f^p_{a,i}$, где $f^p_{a,i}$ являются функциями, отображающими значения других вхождений атрибутов p-того правила вывода в значение вхождения p[i].a . Обозначим через $D^p_{a,i}$ множество аргументов семантического правила $f^p_{a,i}$. Предполагается, что семантические правила локальны в правиле вывода, т.е. для всех правил вывода p из P и для всех $p[i].a \in A(p)$

$$D^p_{a,i} \subseteq A(p).$$

Таким образом, абстрактная атрибутная грамматика AG содержит:

а) регуляризованную КС-грамматику G,

б) множества атрибутов A(X) для всех $X \in V$ и

в) наборы семантических правил для всех правил вывода грамматики G.

Пусть T – дерево вывода программы $w \in L(G)$. Построим

атрибутированное дерево T_1 таким образом, что с каждой вершиной дерева T, порожденной символом $X \in V$, сопоставляется множество A(X), а с каждой вершиной, соответствующей применению p-того правила вывода, сопоставляются все семантические правила этого правила вывода.

Для определения семантики программы w по атрибутированному дереву T_1 вычисляются атрибуты, приписанные вершинам этого дерева и строится семантическое дерево T_2, которое и является семантикой программы w. Дерево T_2 отличается от дерева T_1 тем, что с каждой вершиной сопоставлены только значения семантически существенных атрибутов соответствующего символа.

ААГ применяются в подходе [2] следующим образом. В описании создаваемого языка задаются два представления - конкретный и абстрактный синтаксисы программ этого языка. Во втором представлении из синтаксиса сохраняются только фрагменты, существенные с точки зрения определения семантики языка. Состав и структура этих фрагментов, а также их семантика определяются абстрактной атрибутной грамматикой. Структура атрибутированного дерева конкретной программы определяется на базе дерева вывода.

2. Описание семантики итеративных языковых конструкций

Пусть в атрибутированном дереве T_1 вершина M соответствует применению p-того правила вывода и $p : Y \to \alpha$ содержит в правой части итеративное подвыражение $(p[i] \ldots p[i+k])^+$. Кроме того, пусть среди непосредственных потомков вершины M вершины $M_{j_1}, \ldots, M_{j_l}$ $(l \geq 1)$ - все потомки, соответствующие

символу p[j] этого подвыражения. Вхождению p[j].a в дереве T_1 соответствует совокупность атрибутов a, приписанных вершинам $M_{j_1},\ldots,M_{j_l}$. Таким образом, итеративное вхождение p[j].a понимается как список длиной $l \geq 1$. Элемент этого списка обозначается через p[j].CURRENT.a, первый элемент - p[j].FIRST.a и последний - p[j].LAST.a. Движение по этому списку p[j].a допускается либо слева направо, либо справа налево.

Пусть простое семантическое правило имеет вид

<левая часть> := <выражение> ,

где левая часть и выражение включают вхождения атрибутов и элементы итеративных вхождений атрибутов. Введем для итеративных вхождений семантические правила специального вида. Составное семантическое правило - это заключенная в фигурные скобки последовательность простых семантических правил, левые части которых являются элементами одного и того же итеративного вхождения. Порядок выполнения этих правил определяется порядком их записи. Таким образом, составное семантическое правило определяет значение итеративного вхождения атрибута при выделенном первом и/или последнем элементе этого вхождения.

Итеративное семантическое правило - это семантическое правило, правой частью которого является последовательность выражений, включающих неитеративные вхождения и элементы итеративных вхождений. Эти выражения выполняются в порядке их записи. Итеративное семантическое правило трактуется как примитивно-рекурсивная функция, заданная схемой вычисления в правой части этого правила и определяющая значение неитера-

тивного вхождения атрибута. Другими словами, этот вид семантических правил позволяет использовать значения итеративных вхождений атрибутов. Наличие элемента p[j].CURRENT.a в составе семантического правила или выражения (в итеративном правиле) означает, что это правило (выражение) выполняется неоднократно в зависимости от количества элементов итеративного вхождения p[j].a в атрибутированном дереве.

Пример. Рассмотрим определение объектного кода последовательности операторов, конкретный синтаксис которой приведен на рис. 1.

$$Z \longrightarrow ID := E(;\ ID := E)^*$$
$$E \longrightarrow T(+\ T)^*$$
$$T \longrightarrow ID(\times\ ID)^*$$
$$V_N = \{Z,E,T\},\ V_T = \{ID, :=, ;, +, \times\},\ S_0 = Z .$$

Рис. 1.

Пусть нетерминалы Z, E, T имеют по два атрибута adr и code, терминал ID обозначает идентификатор и имеет единственный атрибут adr. Значением adr служит адрес ячейки памяти, в которой хранится значение соответствующего (под)выражения.

Далее, пусть объектный код содержит операции ASSIGN, ADD и MULT, параметрами которых являются адреса ячеек памяти. Операция ASSIGN(a,b) означает пересылку содержимого ячейки a по адресу b, операция ADD(a,b,c) (MULT(a,b,c)) - пересылку суммы (произведения) значений в ячейках a и b по адресу c.

В абстрактной атрибутной грамматике (рис. 2) запись p[j].PRED.a обозначает левый сосед элемента p[j].CURRENT.a итеративного вхождения p[j].a . Кроме того, символ '||' обо-

$V_N = \{Z,E,T\}$, $V_T = \{ID\}$, $S_0 = Z$

$A(Z) = \{adr,code\}$, $I(Z) = \{adr\}$, $S(Z) = \{code\}$,
$A(E) = \{adr,code\}$, $I(E) = \{adr\}$, $S(E) = \{code\}$,
$A(T) = \{adr,code\}$, $I(T) = \{adr\}$, $S(T) = \{code\}$,
$A(ID) = \{adr\}$, $I(ID) = \emptyset$, $S(ID) = \{adr\}$,

$SEM(Z) = \{adr,code\}$, $EXT(Z) = \{adr\}$,
$SEM(E) = \emptyset$, $EXT(E) = \emptyset$
$SEM(T) = \emptyset$, $EXT(T) = \emptyset$
$SEM(ID) = \{adr\}$, $EXT(ID) = \{adr\}$

1: $Z \longrightarrow (ID\ E)^+$

1.1. {E.FIRST.adr := Z.adr;E.CURRENT.adr := E.PRED.adr + 1}

1.2. Z.code := {E.FIRST.code || 'ASSIGN(E.FIRST.adr,
ID.FIRST.adr)';Z.code || E.CURRENT.code ||
'ASSIGN(E.CURRENT.adr,ID.CURRENT.adr)'}

2: $E \longrightarrow (T)^+$

2.1. {T.FIRST.adr := E.adr;T.CURRENT.adr := T.PRED.adr + 1}

2.2. E.code := {T.FIRST.code;E.code || T.CURRENT.code ||
'ADD(E.adr,T.CURRENT.adr,E.adr)'}

3: $T \longrightarrow (ID)^+$

3.1. T.code := {'ASSIGN(ID.FIRST.adr,T.adr)';
T.code || 'MULT(T.adr,ID.CURRENT.adr,T.adr)}

Рис. 2.

значает конкатенацию цепочек объектного кода и символ '+' – сложение целых чисел. Составными семантическими правилами являются правила 1.1, 2.1, а итеративными – 1.2, 2.2 и 3.1. Значения вхождений Z.adr и ID.adr определяются внешней средой до вычислений на атрибутированном дереве.

3. Вычисление значений атрибутов

В общем случае семантические правила в атрибутированном дереве выполняются в произвольном порядке. Любое семантическое правило можно применять после вычисления значений всех его аргументов (вхождений атрибутов). Для вычисления атрибутов предполагается корректность [3] абстрактной атрибутной грамматики. Точнее, абстрактная атрибутная грамматика называется корректной, если не существует такого атрибутированного дерева в этой грамматике, что вычисление некоторого атрибута в этом дереве оказывается невозможным.

Эффективность реализации атрибутных грамматик прежде всего зависит от выбора алгоритма вычисления атрибутов [4]. В зависимости от алгоритма либо ограничиваются возможности атрибутных грамматик, либо усложняется их реализация. Учитывая конкретный алгоритм вычисления атрибутов, можно достичь более простой и более эффективной проверки корректности. Рассматриваемая реализация ААГ основана на описываемой ниже стратегии вычисления атрибутов [5].

Одновизитной стратегией вычисления атрибутов называется любой способ обхода атрибутированного дерева, при котором каждое поддерево T' посещается не больше одного раза так, что при входе в T' вычисляются некоторые унаследованные атрибуты

корня дерева T' и при выходе из T' – некоторые синтезированные атрибуты его корня. Атрибутная грамматика называется *одновизитной*, если для каждого атрибутированного дерева в этой грамматике существует одновизитная стратегия вычисления всех атрибутов этого дерева.

Пусть для одновизитного вычислителя по каждому правилу вывода p из P определена последовательность визитов $\langle i_1,\ldots,i_{n_p}\rangle$ к символам в правой части этого правила. Далее, пусть $M_1,\ldots,M_n$ – все непосредственные потомки вершины M атрибутированного дерева T_1. В случае обычных атрибутных грамматик каждому символу $p[i]$ в правой части p-того правила соответствует в точности одна вершина M_i, т.е. $n = n_p$. Таким образом, последовательность $\langle i_1,\ldots,i_{n_p}\rangle$ и определяет порядок обхода вершин $M_1,\ldots,M_n$. В случае ААГ возможно, что одному символу $p[j]$ из правой части p-того правила вывода соответствует несколько непосредственных потомков $M_{j_1},\ldots,M_{j_l}$ ($l \geq 1$) вершины M, т.е. количество всех непосредственных потомков превышает n_p. Выбираем для ААГ следующую одновизитную стратегию.

При визите к группе вершин $M_{j_1},\ldots,M_{j_l}$ сначала вычисляются все значения унаследованных атрибутов этих вершин, т.е. значения вхождений $p[i_j].a \in I(p,i_j)$. Затем обходами поддеревьев с корнями $M_{j_1},\ldots,M_{j_l}$ (в указанном порядке) вычисляются все значения синтезированных атрибутов этих вершин, т.е. значения вхождений $p[i_j].a \in S(p,i_j)$.

4. Построение семантического вычислителя

Абстрактная атрибутная грамматика является <u>каноничной</u>, если для всех p из P и для всех $f^p_{a,i}$

$$D^p_{a,i} \subseteq I(p,0) \cup \bigcup_{j=1}^{n_p} S(p,j).$$

Таким образом, в каноничной грамматике множества аргументов семантических правил любого правила вывода не содержат вхождений атрибутов, значения которых вычисляются при этом же правиле вывода. Требование каноничности не является существенным ограничением – как обычную атрибутную грамматику [5] так и корректную абстрактную атрибутную грамматику можно преобразовать в каноничную форму. С другой стороны, свойство каноничности упрощает реализацию (абстрактных) атрибутных грамматик, так как ограничиваются допустимые виды зависимостей между вхождениями атрибутов. В реализации рассматриваются только каноничные ААГ – свойство каноничности проверяется во время построения вычислителя.

Пусть

$$A'(p) = S(p,0) \cup \bigcup_{j=1}^{n_p} I(p,j) \quad \text{и}$$

$$A''(p) = I(p,0) \cup \bigcup_{j=1}^{n_p} S(p,j) \quad .$$

Для каждого p из P построим <u>отношение зависимости</u> $\delta_p \subseteq A'(p) \times A(p)$, где

$$\delta_p = \{(p[i].a, p[j].b) \mid p[i].a \in A'(p) \,\&\, p[j].b \in D^p_{a,i} \,\&\, 0 \le i,j \le n_p\}.$$

Каноничность грамматики AG проверяется условием: для всех правил вывода p из P

$$\delta_p \subseteq A'(p) \times A''(p).$$

Для определения последовательностей визитов построим для всех p из P графы

$$G_p = (N_p, \beta_p),$$

где множество вершин $N_p = \{1,\ldots,n_p\}$, множество дуг $\beta_p \subseteq N_p \times N_p$ и

$$\beta_p = \{(i,j) \mid p[i].a \in S(p,i) \& p[j].b \in I(p,j) \& (p[j].b, p[i].a) \in \delta_p \& 1 \le i,j \le n_p\}.$$

Аналогично классическим атрибутным грамматикам [5], для ААГ имеет место следующее утверждение. Каноничная абстрактная атрибутная грамматика является одновизитной тогда и только тогда, когда графы G_p ациклические для всех правил вывода p из P.

Таким образом, проверка корректности грамматики сводится в данном случае к проверке ацикличности графов G_p. Последовательность визитов $\langle i_1,\ldots,i_{n_p}\rangle$ для p-того правила вывода определяется после проверки корректности топологической сортировкой вершин графа G_p.

При вычислении атрибутов встает проблема распределения памяти значениям вхождений атрибутов. Значения вхождения $p[i_j].a$ – это вектор (длиной $l \ge 1$) значений атрибутов a вершин $M_{j_1},\ldots,M_{j_l}$. Пусть (c,l) – адрес значения, где c – адрес вектора в памяти и l – его длина. В одновизитном вычислителе используется магазинное распределение памяти. Адреса значе-

ний определяются в процессе вычисления системой распределения памяти и они доступны до тех пор, пока находятся в стэках вычислителя.

В заключение приведем (на рис. 3) процедуру одновизитного вычисления атрибутов в поддереве T' атрибутированного дерева T_1. Пусть A - некоторое множество вхождений атрибутов и x - стэк, элементы которого - адреса значений. Далее, пусть операция push (x,A) запишет в стэк x адреса значений элементов A, операция pop (x,A) удаляет из x эти же адреса и операция save (M) перенесет адреса семантически существенных атрибутов вершины M из стэков в соответствующую вершину семантического дерева T_2. Кроме того, пусть INH и SYN - стэки вычислителя, элементами которых являются адреса значений вхождений унаследованных и синтезированных атрибутов соответственно.

```
procedure evaluate (M)
begin
for j := 1 to n_p do
    push (INH, I(p,i_j)); push (SYN, S(p,0));
    вычислить значения элементов I(p,i_j);
    for m := 1 to l do
        evaluate (M_jm); save (M_jm)
                    od;
    pop (INH, I(p,i_j))
                  od;
    вычислить значения элементов S(p,0);
for j := 1 to n_p do
    pop (SYN, S(p,i_j))
                  od
end
```

Рис. 3.

Литература

1. Вооглайд А.О., Мериете М.Б. Абстрактные атрибутные грамматики, Программирование, 1982, № 5, 17-27.
2. Вооглайд А.О., Мериете М.Б. О технологии применения атрибутных систем построения трансляторов, сб. 350 лет математики в Тартуском университете (тезисы докл.), Тарту, 1982, 91-93.
3. Кнут Д.Э. Семантика контекстно-свободных языков, сб. Семантика языков программирования, ред. В.М.Курочкин, М., "Мир", 1980, 137-161.
4. Мериете М.Б. Методы реализации атрибутных схем в системах построения трансляторов, Программирование, 1980, № 5, 40-49.
5. Engelfriet, J., Filé, G. Formal properties of one-visit and multi-pass attribute grammars. Proc. of the 7th Coll. on Automata, Lang. and Progr., 1980, 182-194.
6. Giegerich, R., Wilhelm, R. Attribute evaluation, Le Point sur la Compilation, Cours de la Commission des Communautés Européennes, M. Amirchahy and D. Neel (eds.), IRIA, 1978, 337-365.
7. Kennedy, K., Warren, S.K. Automatic generation of efficient evaluators for attribute grammars. Third ACM Symp. on Princ. of Progr. Lang., 1976, 32-49.
8. Lewi, J., DeVlaminck, K., et al. A programming methodology in compiler construction, North Holland, 1979.
9. Räihä, K.-J. Experiences with the compiler writing system HLP, Lect. Notes Comp. Sci., 1980, 94, 350-362.

О СТАБИЛЬНОСТИ МОДЕЛИ ЛИНЕЙНОГО РЕГРЕССИОННОГО АНАЛИЗА ПРИ БОЛЬШОМ ЧИСЛЕ РЕГРЕССОРОВ

Э.-М. Тийт

0. Целью настоящей статьи является исследование асимптотического распределения эмпирических регрессионных моделей при некоторых предположениях о теоретической регрессионной модели. В результате этого возникает возможность оценить величину смещения множественного коэффициента корреляции теоретически наилучшей модели. Это смещение возникает в результате необходимости оценить значение наилучшей модели по т.н. "эмпирически наилучшей" модели, которая, как правило, сильно смещена.

1. Предполагаем, что теоретическая корреляционная матрица между регрессандом Y и регрессорами $X_1,\dots,X_m$ имеет форму

$$R = \left(\begin{array}{c|c} 1 & \beta' \\ \hline \beta & R(X) \end{array}\right), \tag{1}$$

где $\beta' = (\beta_1,\dots,\beta_m)$ – вектор корреляции $r(Y,X_1)$, а

$$R(X) = R_m(1,\alpha) = (r(X_i,X_j)),$$

где

$$r(X_i,X_j) = \begin{cases} \alpha, & \text{если } i \neq j, \\ 1, & \text{если } i = j. \end{cases} \tag{2}$$

В таком случае, как хорошо известно (см. напр. [3], стр. 69) обратная матрица $[R(X)]^{-1}$ имеет форму $R_m(a,c)$, где

$$\begin{cases} c = -\dfrac{\alpha}{J(1,\alpha,k)(1-\alpha)}, \\ a = \dfrac{J(1,\alpha,k-1)}{J(1,\alpha,k)(1-\alpha)}. \end{cases} \tag{3}$$

Здесь введено обозначение

$$J(x,y,u) = x + (u-1)y . \tag{4}$$

Из формулы (3) вытекает, что если выполнены условия (1) и (2), то коэффициент множественной корреляции к выражается из формулы[1] (см. [4,5]):

$$K^2 = \frac{m\bar{\beta}^2}{J(1,\alpha,m)} + \frac{mD\beta}{1-\alpha}, \tag{5}$$

где

$$\bar{\beta} = \frac{1}{m}\sum_{i=1}^{m}\beta_i; \qquad D\beta = \frac{1}{m}\sum_{i=1}^{m}(\beta_i-\bar{\beta})^2. \tag{6}$$

В частном случае, когда

$$\beta_i = \beta \qquad (i=1,\ldots,m), \tag{7}$$

то формула (5) приобретает более простой вид

$$K^2 = \frac{m\beta^2}{J(1,\alpha,m)} . \tag{8}$$

Полной моделью называется линейное регрессионное соотношение

$$Y = \sum_{i=1}^{m} b_i X_i .$$

---

[1] Здесь и в дальнейшем предполагается, что соответствующие модели существуют, т.е. $0 \le K \le 1$.

Нас интересуют в основным неполные модели, которые идентифицируются выбором регрессоров при помощи индекс-множества I:

$$I = \{i_1,\dots,i_k\},\quad 1 \le i_j < i_{j+1} \le m \ (j=1,\dots,k-1;\ 1 \le k < m).$$

Коэффициент множественной корреляции, соответствующий модели I, обозначается символом K(I):

$$K^2(I) = \frac{k[\bar\beta(I)]^2}{J(1,\alpha,k)} + \frac{kD\beta(I)}{1-\alpha}, \qquad (5')$$

где

$$\bar\beta(I) = \frac{1}{k}\sum_{j=1}^{k}\beta_{i_j},\quad D\beta(I) = \frac{1}{k}\sum_{j=1}^{k}(\beta_{i_j} - \bar\beta(I))^2, \qquad (6')$$

а в частном случае (7)

$$K^2(I) = \frac{k\beta^2}{J(1,\alpha,k)}. \qquad (8')$$

Называем модели, содержащие k регрессоров, в дальнейшем k-моделями.

2. Рассмотрим прежде всего случай, когда выполнено условие (7). Тогда все теоретические k-модели равны друг другу, притом их число равняется $N = \binom{k}{m}$. Например, в случае $m = 50$, $k = 10$ $N = \binom{10}{50} = 1.0272\cdot10^{10}$. Обозначаем множество всех k-компонентных индекс-множеств через $\mathfrak{I}_k$.

Предполагаем, что имеется выборка объема n из генеральной совокупности, удовлетворяющей условиям (1), (2) и (7), притом n "достаточно большое".

Как известно, выборочный коэффициент множественной корреляции $\hat K(n)$ имеет при предположении нормальности исходного распределения асимптотически нормальное распределение

$$\hat{K}(n) \sim N(K, \frac{(1-K^2)^2}{n}). \qquad (9)$$

При наших предположениях мы можем явно вычислить параметры асимптотического распределения (9) для случайного представителя $\hat{K}(I,n)$ множества всех выборочных k-моделей на основании выборки объема n:

$$\begin{cases} E\hat{K}(I,n) = \dfrac{\sqrt{k}\,|\beta|}{\sqrt{J(1,\alpha,k)}}, \\ \sigma\hat{K}(I,n) = \dfrac{J(1,\alpha,k)-k\beta^2}{\sqrt{n}\,J(1,\alpha,k)} = \dfrac{J(1,\alpha-\beta^2,k)+\beta^2}{\sqrt{n}\,J(1,\alpha,k)}. \end{cases} \qquad (10)$$

Эмпирически оптимальной моделью является модель I^*, определенная условием:

$$\hat{K}(I^*,n) = \max_{I \in \mathfrak{I}_k} \hat{K}(I,n)$$

для данной выборки. Такая модель на практике, как правило, разыскивается путем различных пошаговых процедур.

Для оценивания "действительного качества" имеющейся модели необходимо оценить смещение:

$$E\hat{K}(I^*,n) - K(I) = E\hat{K}(I^*,n) - \frac{\sqrt{k}\,|\beta|}{\sqrt{J(1,\alpha,k)}}.$$

Для этого надо оценить величину $E\hat{K}(I^*,n)$.

3. Для исследования распределения $\hat{K}(I^*,n)$ сделаем пока одно упрощающее предположение: предполагаем, что все выборочные оценки $\hat{K}(I,n)$, вычисленные по той же выборке, но по разным моделям, независимые. Это предположение неоправдано (см. [2]), так как даже при непересекающихся комплектов регрессоров они зависят от того же самого регрессанда. Но в случаях, представляющих практический интерес, корреляции между моде-

лями достаточно слабы и ими возможно в первом приближении пренебрегать.

При таком дополнительном предположении легко найти распределение максимума из всех $\hat{K}(I,n)$, т.е. эмпирически оптимальной модели $\hat{K}(I^*,n)$, которую мы в дальнейшем для простоты обозначаем через K^*:

$$P(K^* < x) = P(\prod_{i=1}^{N} \hat{K}_i < x) = P[\hat{K}(I,n) < x]^N,$$

где через $\hat{K}_i$ обозначены эмпирические модели $\hat{K}(I,n)$, упорядоченные некоторым фиксированным образом.

В качестве оценки для коэффициента K^* пользуемся его медианой M, которая просто выражается из полученной формулы:

$$P(K^* < M) = 0.5 \Longrightarrow P(\hat{K} < M) = 0.5^{1/N},$$

где

$$M = K(I) + Q \cdot \sigma(\hat{K}),$$

Q – квантиль стандартизированного нормального распределения, соответствующий вероятности $0.5^{1/N}$. Величину Q можно найти из таблиц (см. напр. [1], стр. 12–13) или при слишком больших значений N вычислить по приближенным формулам. В настоящей работе пользовались итерационной формулой, выведенной из формулы [1], стр. 11:

$$Q^2 = 2\ln R(Q) + 2\ln N - C, \qquad (11)$$

где R(Q) – цепная дробь,

$$R(x) = \cfrac{1}{1 + \cfrac{1}{x + \cfrac{2}{x + \ldots}}},$$

а $C = \ln 2\pi + 2\ln\ln 2$. Так как распределение $\hat{K}(I,n)$ асимптотически стремится к симметричному, то асимптотическая оценка

смещения эмпирически оптимальной модели (при оценивании с помощью медианы) равняется величине

$$Q(N) \cdot \sigma(\hat{K}) ,$$

где $Q(N) = Q$ найдется из формулы (11), а $\sigma(\hat{K}) = \sigma\hat{K}(I,n)$ – из формулы (10).

4. ПРИМЕР 1

Рассмотрим исходную матрицу с параметрами $m = 50$, $\alpha = 0.2$, $\beta = 0.4$ (см. формулы (1) и (2)).

Предполагаем, что требуется рассмотреть k–модели при $k = 10$. В таком случае $N = 1.0272 \cdot 10^{10}$ и $Q(N) = 6.4216$. Истинное значение множественного коэффициента корреляции любой 10–модели, вычисленное по формуле (8), равняется $K_{10} = 0.75593$, а дисперсия выборочного коэффициента корреляции при объеме выборки n есть $\sigma\hat{K}(n) = \frac{0.42857}{\sqrt{n}}$. По этим данным легко найти оценку коэффициента множественной корреляции наилучшей эмпирической модели K^* и оценить смещение $\hat{b}_n$, возникающее в результате применения наилучшей эмпирической модели. Соответствующие результаты заданы в таблице 1.

Отсюда видно, что ввиду сверхбольшого числа N теоретически эквивалентных k–моделей смещение $\hat{b}_n$ несколько раз превышает обыкновенную "выборочную ошибку" $t\sigma_n(\hat{K})$ оценки коэффициента корреляции $\hat{K}$. Заметим, что "истинное значение" множественного коэффициента корреляции при полной модели равняется 0.861 . Так как всегда $K(I) \leq K$, притом это неравенство сохраняется при всех моделях, то можно предполагать, что значения смещения, заданные в таблице 1, переоценены. Кроме вышеупомянутой зависимости отдельных моделей может вызвать

смещение оценок еще несправедливое предположение симметричности распределения $\hat{K}$. Чтобы проверять это, вычислим еще некоторые квантили распределения $\hat{K}$, пользуясь формулой (11'):

$$x_q^2 = 2\ln N + 2\ln R(x_q) - C(q), \qquad (11')$$

где $C(q) = \ln 2\pi + 2\ln\ln(1/q)$, и так как равенство $x_q + x_{1-q} \approx x_{0.5}$ имеет практически точно место для разных значений q (например, $x_{0.9} + x_{0.1} = 6.469$), то эффект несимметричности распределения $\hat{K}$ весьма несущественный.

5. Рассмотрим случай изменяющихся k-моделей. Простейшей возможностью для этого является "дихотомическая модель" в смысле коэффициентов β:

$$\beta_i = \begin{cases} \beta, & \text{если } i=1,\dots,q, \\ 0, & \text{если } i=q+1,\dots,k, \end{cases}$$

притом предполагается $\min(q, m-q) \geq k$. Тогда каждая k-модель характеризуется числом s т.н. β-регрессоров X_i $(r(X_i, Y) = \beta)$. Соответствующие модели называются (k,s)-моделями; их число при наших предположениях равняется

$$N_s = \binom{q}{s}\binom{m-q}{k-s},$$

притом естественно $\sum_{s=0}^{k} \binom{q}{s}\binom{m-q}{k-s} = N$.

В рассматриваемом случае $\bar{\beta} = \frac{s}{k}\beta$ и $D\beta = \frac{s(k-s)}{k^2}\beta^2$. Подставляя эти значения в формулу (5'), имеем:

$$K^2(1_{k,s}) = \frac{s}{k}\beta^2\left[\frac{s}{J(1,\alpha,k)} + \frac{k-s}{1-\alpha}\right]. \qquad (12)$$

Таким образом, выборочное распределение P_n k-модели можно характеризовать как смесь, состоящая из (k+1) нормальных

распределений:

$$P_n = \sum_{s=0}^{k} p_s N(\mu_s, \sigma_{s,n}), \qquad (13)$$

где $p_s = N_s/N = \binom{q}{s}\binom{m-q}{k-s}\cdot N^{-1}$,

$$\mu_s = \beta\left\{\frac{s}{k}\left[\frac{s}{J(1,\alpha,k)} + \frac{k-s}{1-\alpha}\right]\right\}1/2 ,$$

$$\sigma_{s,n} = \frac{1}{\sqrt{n}}\left\{1 - \frac{s}{k}\beta^2\left[\frac{s}{J(1,\alpha,k)} + \frac{k-s}{1-\alpha}\right]\right\}.$$

Из выражения (12) легко найти значение s, максимизирующее K^2:

$$\frac{\partial}{\partial s} K^2(k,s) = C\,\frac{\partial}{\partial s}\left\{s^2(1-\alpha) + s(k-s)J(1,\alpha,k)\right\} \Rightarrow 0$$

$$s = \frac{J(1,\alpha,k)}{2\alpha}$$

и

$$s^* = [s + 0.5] . \qquad (14)$$

Ввиду непрерывности $K^2(k,s)$ как функции от s следующей по величине $K(I_{k,s})$ является соответственно $k,(s^*+1)$- или $k^*,(s-1)$-модель (или обе).

Отсюда вытекает вывод, что для исследования поведения эмпирически оптимальной модели необходимо рассмотреть поведение оценки выборочного максимума (k,s^*)-модели и близких к ней моделей методом, описанным в предыдущем пункте.

6. ПРИМЕР 2

Рассмотрим пример со следующими параметрами: m=50, q=25, $\alpha = 0.2$, $\beta = 0.4$, притом требуют построить k-модель, k=10 и рассматривают значения n=50, 100, 200, 500, 1000, 2000, 5000 и 10 000.

Прежде всего выпишем характеристики распределения P_n, т.е. численности отдельных моделей $\varkappa(s)$, их весы в смеси (13)

$\varkappa(s)/N$, теоретические средние $\mu_s = K(I_{s,k})$ и стандартные отклонения $\sigma_{s,n} = \frac{1}{\sqrt{n}}(1 - K^2(I_{s,k}))$.

В данном примере $s^* = 7$, а по значению K^2 совпадают соответственно модели при $s=6$ и $s=8$, $s=5$ и $s=9$ да $s=4$ и $s=10$. Вычислим для каждого значения s теоретическую медиану M выборочного максимума при разных значениях n (см. табл. 3).

Так как разные модели имеют разную частоту в смеси (13), то их эмпирические максимумы расположены на разные расстояния от теоретических средних $\mu(s)$, и в результате эмпирически наилучшей моделью (при маленьких и средних объемах выборки) окажется не теоретически наилучшая модель, а некоторая другая (см. последний столбец таблицы). Заметим, что при $n = 50$ все эмпирические коэффициенты корреляции, заданные в таблице 3, вырождены.

Пользуясь формулой (11) вычислены и вероятности событий $K_n(I_s) > K_n(I_{s^*})$, т.е. событий, когда случайно выбранная эмпирическая (k,s)-модель является больше случайно выбранной (k,s^*)-модели, где k фиксирован (в данном случае $k = 10$) и s^*

Таблица 4.

Вероятности событий $K_n(I_s) > K_n(I_{s^*})$ при разных объемах выборки.

s \ n	50	100	200	500	1000	2000	5000
6, 8	0.444	0.422	0.390	0.329	0.265	0.188	0.081
5, 9	0.297	0.227	0.145	0.047	0.009	0.0005	0
4,10	0.138	0.062	0.015	0.0005	0	0	0
3	0.04	0.007	0.0005	0	0	0	0
2	0.006	0.0005	0	0	0	0	0
1	0.0005	0	0	0	0	0	0

обозначает теоретически наиболее подходящее число β-регрессоров (в рассматриваемом случае $s^*=7$). Результаты вычислений заданы в таблице 4.

7. Еще более интересным для практики является случай, когда исходная корреляционная матрица (1) имеет блочную структуру (1'):

$$R = \begin{pmatrix} 1 & \beta_1' & \cdots & \beta_r' \\ \beta_1 & A_{11} & \cdots & G_{1r} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_r & G_{r1} & \cdots & A_{rr} \end{pmatrix}, \qquad (1')$$

где диагональные блоки имеют диагональную структуру: $A_{ll} = (a^l_{ij})$,

$$a^l_{ij} = \begin{cases} 1, & \text{если } i = j \\ \alpha^l, & \text{если } i \neq j \end{cases} \quad (i,j=1,\ldots,k_l),$$

а внедиагональные блоки постоянны: $G_{hl} = (\gamma^{hl}_{ij})$, $\gamma^{hl}_{ij} = \gamma^{hl}$ $(i=1,2,\ldots,k_h; l=1,2,\ldots,k_l)$. Размерность матрицы R есть $1+m$, где $m = \sum_{l=1}^{r} k_l$, $r > 1$.

В таком случае множественный коэффициент корреляции вычисляется по формуле:

$$K^2 = b' G^{-1} b + \sum_{l=1}^{r} \frac{k_l}{\alpha^l} D\beta_l, \qquad (15)$$

где $b = (\sqrt{k_1}\bar{\beta}_1, \ldots, \sqrt{k_r}\bar{\beta}_r)$,

$$\bar{\beta}_l = \frac{1}{k_l} \sum_{i=k+1}^{k+k_l} \beta_i, \qquad D\beta_l = \frac{1}{k_l} \sum_{i=k+1}^{k+k_l} (\beta_i - \bar{\beta}_l)^2,$$

$$k = \sum_{i=1}^{l-1} k_i, \qquad l = 1,\ldots,r,$$

а G – $(r \times r)$-матрица с элементами g_{ij}:

$$g_{ij} = \begin{cases} J(1,\alpha^1,k_1), & \text{если } i = j, \\ \sqrt{k_1 k_j}\,\gamma^{1j}, & \text{если } i \neq j. \end{cases}$$

Соответственно G^{-1} – обратная матрица матрицы G, которая положительно определена тогда и только тогда, когда соответствующая модель существует.

Если вектор β – дихотомический, состоящий из нулевых и ненулевых значений, то формула (15) упрощается:

$$\bar{\beta}_1 = \frac{s_1}{k_1}\beta, \quad D\bar{\beta}_1 = \frac{s_1(k_1-s_1)}{k_1^2}\beta^2, \qquad (15')$$

где s_1 $(0 \le s_1 \le k_1)$ – число β-регрессоров в блоке β_1. В таком случае модель I_k определяется $2r$-мерным вектором $(p_1,q_1,\ldots,p_r,q_r)$, притом должны быть выполнены условия: $0 \le p_i \le s_i$; $0 \le q_i \le k_i - s_i$ $(i=1,\ldots,r)$ и $\sum_{i=1}^{r} p_i + \sum_{i=1}^{r} q_i = k$. В таком случае p_i показывает число β-регрессоров и q_i – число 0-регрессоров в i-м блоке $(i=1,\ldots,r)$. Для каждого k число разных моделей вычисляется просто:

$$N_k = \prod_{i=1}^{r} \binom{p_i}{s_i}\binom{q_i}{k_i-s_i},$$

а при любом объеме выборки n асимптотическое распределение выборочного коэффициента корреляции $\hat{R}_n$ есть следующая смесь:

$$P_n = \sum_{p_1=0}^{s_1^0} \sum_{q_1=0}^{k_1^0} \cdots \sum_{p_r=0}^{s_r^0} \binom{p_1}{s_1^0}\binom{q_1}{k_1^0} \cdots \binom{p_r}{s_r^0}\binom{q_r}{k_r^0} N(\mu(I_k),\sigma_n(I_k)),$$

где $s_1^0 = \min(k,s_1)$; $k_1^0 = \min(k-s_1^0,k_1-s_1)$, ... ;

$s_r^0 = \min\left(k-\sum_{i=1}^{r-1}(s_i^0+k_i^0),s_r\right)$ и $k_r^0 = k-\sum_{i=1}^{r} s_i^0 - \sum_{i=1}^{r-1} k_i^0$,

а $\mu(I_k)$ вычисляется из формулы (15'), подставляя вместо s_i

соответственно p_i и вместо k_i суммы $p_i + q_i$ $(i=1,\ldots,r)$. По найденному коэффициенту корреляции стандартным образом вычисляется и $\sigma_n = \frac{1}{\sqrt{n}}(1-k^2)$.

8. ПРИМЕР 3

Рассмотрим пример: пусть исходная матрица имеет следующий вид

	1	15		15		15		15	
1	0.4	...	0.4	0 ... 0	0.4	...	0.4	0	... 0
0.4	1	...	0.3	0.3 ... 0.3	0.1	...	0.1	0.1	... 0.1
. .	.	.	.	. . . . .	.	.	.	.	. . .
0.4	0.3	...	1	0.3 ... 0.3	0.1	...	0.1	0.1	... 0.1
. .	.	.	.	. . . . .	.	.	.	.	. . .
0	0.3	...	0.3	0.3 ... 1	0.1	...	0.1	0.1	... 0.1
0.4	0.1	...	0.1	0.1 ... 0.1	1	...	0.2	0.2	... 0.2
. .	.	.	.	. . . . .	.	.	.	.	. . .
0.4	0.1	...	0.1	0.1 ... 0.1	0.2	...	1	0.2	... 0.2
. .	.	.	.	. . . . .	.	.	.	.	. . .
0	0.1	...	0.1	0.1 ... 0.1	0.2	...	0.2	0.2	... 1

Необходимо определить наилучшую 10-модель.

Соответствующие вычисления сделаны на миникалькуляторе "APPLE".

На рис. 1 представлено теоретическое распределение случайно выбранной k-модели (функция плотности и функция распределения). Общее число моделей $N = 7.54 \cdot 10^{10}$, число различных моделей – 286. Теоретически наилучшей моделью оказалось модель, где $p_1=4$, $q_1=3$, $p_2=3$ и $q_2=0$. Значит, в наилучшую модель

входит 4 регрессоров X_i из первого блока ($r(Y,X_i) = 0.4$, $r(X_i,X_j) = 0.3$), три некоррелированных с Y регрессоров второго блока ($r(Y,X_i) = 0$, $r(X_i,X_j) = 0.3$) и три регрессоров третьего блока ($r(Y,X_i) = 0.4$, $r(X_i,X_j) = 0.2$).

Для исследования поведения эмпирических моделей пользовались методом статистического моделирования: из N моделей выбрали случайно ν моделей, и для каждого из них вычислили случайный эмпирический представитель (при предположении, что объем выборки равняется заданному значению n). Затем исследовали поведение полученного эмпирического распределения, где вероятность каждого пункта равняется $1/\nu$. На рис. 2 представлен случай, когда $n=10$, $\nu=100$. Понятно, что в таком случае все эмпирические модели вырождены, выбор моделей для практики не представляет интереса. Хотя в данном случае вышеописанная процедура моделирования эмпирических коэффициентов корреляции некорректна, общий результат не противоречит теории: многие модели вырождены.

В случае $n=100$, $\nu=100$ (см. рис. 3) результат представляет уже практический интерес. В данной выборке все модели невырождены, хотя теоретически можно получить и вырожденные (видно, что теоретические эмпирические максимумы десяти наилучших моделей все больше единицы). Заметим и то, что определенный по выборке НЭМ не соответствует ни одной из десяти наилучших моделей и переоценивает наилучшую модель. Значение смещения оценки множественного коэффициента корреляции наилучшей модели равняется $0.959 - 0.888 \approx 0.07$.

По объему выборки $n=1000$ ($\nu=100$) все эмпирические модели невырождены. Видно (см. рис. 4), что и порядок эмпирически

наилучших моделей (найденных по их максимумам) в основном соответствует порядку теоретических моделей, притом смещение порядка 0.01-0.02 . Понятно, что на основании ста случайно выбранных моделей невозможно найти теоретически наилучшей модели, но интересно то, что наилучшая (по данной выборке) модель переоценивает теоретически наилучшую модель только на 0.002 .

Видно, что в этом случае и эмпирическое распределение коэффициентов корреляции сравнительно близко теоретическому распределению (рис. 1).

Литература

1. Оуэн Д.Б., Сборник статистических таблиц. М., 1966.
2. Парринг А.-М., Общее асимптотическое распределение неполных множественных коэффициентов корреляции. Уч.зап.ТГУ, 1980, вып. 541, 18-26.
3. Рао С.Р., Линейные статистические методы и их применения. М., 1968.
4. Тийт Э., Выбор моделей в линейном регрессионном анализе. Труды ВЦ ТГУ, 1981, № 46, 60-84.
5. Тийт Э.-М., Класс допустимых эмпирических моделей в линейном регрессионном анализе. - В сб. Машинные методы обнаружения закономерностей. Новосибирск, 1981, 99-106.

Таблица 1.

Объем выборки n	50	100	200	500	1000	2000	5000	10000	∞
НЭМ K_n^*	вырождено	вырождено	0.950	0.879	0.843	0.817	0.795	0.783	0.756
Смещение $\hat{b}_n$	0.389	0.275	0.195	0.123	0.087	0.062	0.039	0.028	0

Таблица 2.

s	0	1	2	3	4	5	6	7	8	9	10
$N_s/10^6$	3.27	51.07	324.47	1105.7	2240.3	2822.8	2240.3	1105.6	324.47	51.07	3.27
p(s)	.00032	.00497	.03153	.10763	.21810	.27480	.21810	.10763	.03153	.00497	.00032
$\mu(s)$	0	.4309	.5856	.6866	.7559	.8017	.8281	.8367	.8281	.8017	.7559
$\sigma_{s,n}$	$\frac{1}{\sqrt{n}}$	$\frac{.8143}{\sqrt{n}}$	$\frac{.6571}{\sqrt{n}}$	$\frac{.5286}{\sqrt{n}}$	$\frac{.4286}{\sqrt{n}}$	$\frac{.3572}{\sqrt{n}}$	$\frac{.3143}{\sqrt{n}}$	$\frac{.3}{\sqrt{n}}$	$\frac{.3143}{\sqrt{n}}$	$\frac{.3572}{\sqrt{n}}$	$\frac{.4286}{\sqrt{n}}$

Таблица 3.

Характеристики распределений выборочных множественных коэффициентов корреляций при разных объемах выборок

s		7	6 и 8	5 и 9	4 и 10	3	2	1	НЭМ
N_s		$1.1056 \cdot 10^9$	$2.5648 \cdot 10^9$	$2.8739 \cdot 10^9$	$2.2436 \cdot 10^9$	$1.1056 \cdot 10^9$	$3.2447 \cdot 10^8$	$5.107 \cdot 10^7$	
Q_s		6.0732	6.2069	6.2250	6.1858	6.0732	5.8733	5.5589	
n=50	K_n^*	1.0944	1.1040	1.11606	1.1308	1.1406	1.1313	1.0711	s = 3 K_n^*= 1.1406
	$\sigma_{s,n}$	0.0424	0.0444	0.0505	0.0606	0.0748	0.0929	0.1152	
n=200	K_n^*	0.9655	0.9660	0.9589	0.9434	0.9136	0.8585	0.7511	s=6 и 8 K_n^*= 0.9660
	$\sigma_{s,n}$	0.0212	0.0222	0.0253	0.0303	0.0374	0.0465	0.0576	
n=500	K_n^*	0.9182	0.9153	0.9011	0.8745	0.8302	0.7582	0.6333	s = 7 K_n^*= 0.9182
	$\sigma_{s,n}$	0.0134	0.0141	0.0160	0.0192	0.0236	0.0294	0.0364	

Здесь N_s – число соответствующих моделей, а Q_s – значение теоретической медианы максимума из выборки объема N_s, если исходное распределение – стандартизированное нормальное. N_s найдено по формуле (11).

```
MATRIX SIZE=15+15+15+15
CORRELATIONS IN DIAG. BLOCKS .3 AND .2 OUT OF DIAG. .1
CORR. BETWEEN Y AND X ARE 0 AND .4
 MODEL SIZE=10 NO OF ALL MODELS=7.53940276E+10
PSEUDOMAX=0 NO OF SING. MODELS=0
 NO OF DIFFERENT MODELS=286 NO OF REGULAR MODELS=286 SUM PROBABILITY=.999999998
BEST MODEL 4 3 3 0
```

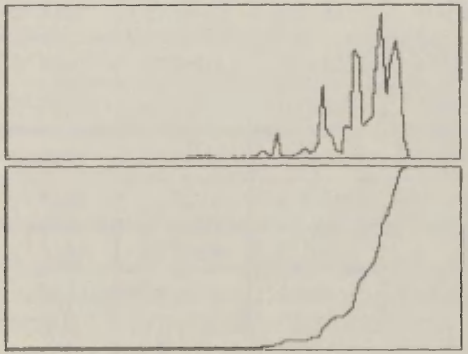


Рис. 1.

```
SAMPLE SIZE=10
10 THEORETICALLY BEST MODELS AND THEIR THEOR. SAMPLE MAXIMUMS
224             .888032789      1.73823428
248             .886495798      1.72550198
220             .880729182      1.74731118
245             .88036818       1.74032386
186             .879843162      1.78923162
219             .879520456      1.79005111
227             .8784928        1.75264128
192             .876578039      1.76649852
180             .872534362      1.77810473
179             .871890255      1.80890335

TOO LITTLE ONES 0 TOO BIG ONES=19 PROBABILITIES ARE 0 AND .19
MIN SAMPLE CORR=.319285838 RESP. THEOR. CORR=.685 MAX SAMPLE CORR=1.20236572 RESP. THEOR. CORR=.
695
```

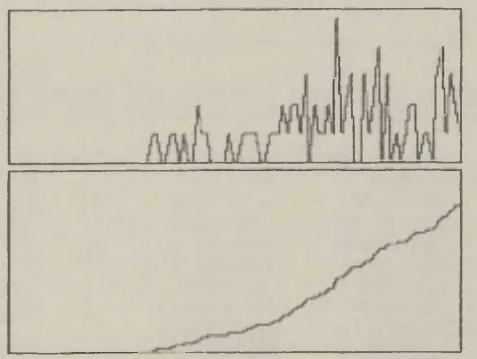



Рис. 2.

SAMPLE SIZE=100
10 THEORETICALLY BEST MODELS AND THEIR THEOR. SAMPLE MAXIMUMS

224	.888032789	1.15689011
248	.886495798	1.15181285
220	.880729182	1.15476647
245	.88036818	1.15231004
186	.879843162	1.16741704
219	.879520456	1.16745553
227	.8784928	1.15492282
192	.876578039	1.15799561
180	.872534362	1.15890086
179	.871890255	1.16819981

TOO LITTLE ONES 0 TOO BIG ONES=0 PROBABILITIES ARE 0 AND 0
MIN SAMPLE CORR=.0684613914 RESP. THEOR. CORR=.415 MAX SAMPLE CORR=.959474363 RESP. THEOR. CORR=
.845

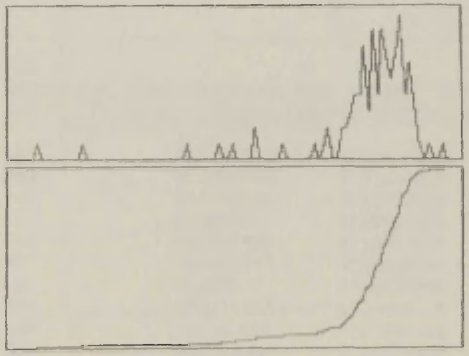


Рис. 3.

SAMPLE SIZE=1000
10 THEORETICALLY BEST MODELS AND THEIR THEOR. SAMPLE MAXIMUMS

224	.888032789	.973052938
248	.886495798	.970396416
220	.880729182	.967387382
245	.88036818	.966363749
186	.879843162	.970782008
219	.879520456	.970573521
227	.8784928	.965907648
192	.876578039	.965570087
180	.872534362	.963091399
179	.871890255	.965591564

TOO LITTLE ONES 0 TOO BIG ONES=0 PROBABILITIES ARE 0 AND 0
MIN SAMPLE CORR=.422094592 RESP. THEOR. CORR=.415 MAX SAMPLE CORR=.890560184 RESP. THEOR. CORR=.
855

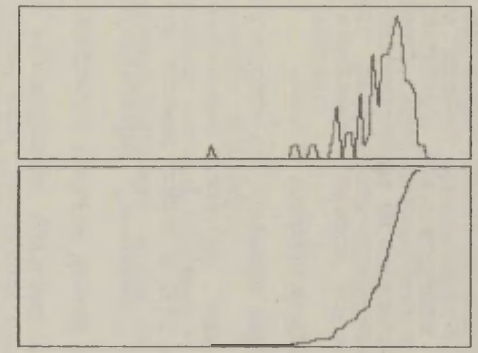


Рис. 4.

О РЕЗУЛЬТАТИВНОСТИ СТАТИСТИЧЕСКОГО АНАЛИЗА ДАННЫХ

Л.М. Тоодинг

1. Характеристика используемой эмпирической совокупности

1.1. Результативность научного исследования, в ходе которого использовались методы прикладной статистики, в существенной мере определяется возможностями применяемой методики анализа данных. Следует учитывать не только теоретическую сущность методов, используемую вычислительную технику, но и организацию процесса обработки – все основные компоненты службы статистической обработки. Анализ службы обработки в целях повышения ее эффективности является, таким образом, неизбежным для успешного выявления содержательной информативности обрабатываемых данных.

Следуя принципу, по которому анализ информационной системы должен протекать независимо от инструментальной части системы [4], в данно* заметке рассматриваются некоторые формальные показатели, характеризующие результативность задач анализа данных научных исследований. Естественно, что богатство эмпирической совокупности данных полностью оцениваемо исключительно специалистами соответствующей области исследования, но можно предполагать и наличие принципиальных, общих

для задач разных областей показателей результативности.

Основой данной заметки являются задачи статистического анализа данных, решенные в Вычислительном центре Тартуского государственного университета за период 1974–1982 при помощи системы статистической обработки данных [1,2] на ЭВМ "Минск-32" и ЭВМ ЕС. Системой до сих пор обслужено свыше 750 задач, притом эксплуатация ее продолжается, также и разработка дополненной версии системы.

Ниже рассматриваются результаты опроса пользователей службой обработки (клиентов), с которыми имелось сотрудничество. Дополнением к настоящей является статья [3], посвященная эмпирической характеристике реального потока задач статистической обработки данных.

1.2. Опрос клиентов проводился в виде анкеты, ниже приводятся данные 90 анкет. В анкете имелись следующие рубрики:

- характеристика клиента по "личным" показателям: трудовой стаж, пол, специальность, научная квалификация;
- структура данных и задачи: размерность статистической выборки, использованные методы;
- мотивы обращения к ВЦ;
- значение консультаций в службе обработки;
- использование методологических пособий системы;
- способы формализации содержательной задачи для системы и интенсивность анализа;
- результативность решения.

Анализ анкетных данных проводился при помощи системы статистической обработки. Рассматривались общие распределе-

ния признаков, выяснились взаимосвязи между признаками перечисленных групп, также применялись факторный и регрессионный анализы. Целью анализа являлось:

- установление по оценке клиента степени эффективности применяемой в ВЦ ТГУ службы статистической обработки;
- исследование целесообразности ряда организационных мер службы;
- выяснение факторов, действующих на результативность обработки.

Главной из названных целей считалась, естественно, последняя.

Результаты анализа используются в дальнейшей разработке системы и внедряются в службу обработки.

1.3. Опрошенные анкетным путем клиенты в большинстве были преподаватели (38%) или научные сотрудники (52%), из них половина имела научную степень. Женщины составляют 40%. Средний трудовой стаж исследуемых 12 $\pm$ 2 лет[1]. До рассматриваемой задачи имели опыт статистического анализа данных больше 60% от всех клиентов. Ни один из опрошенных не отрицал необходимости обращения в ВЦ при исследовании.

Отзывы на результативность в основном положительны; единогласно утверждается, что углублялось убеждение в необходимости применения методов прикладной статистики в их области исследования. По специальностям 1/5 доля задач относится к социологии, 1/4 - к психологии и педагогике, столько же к медицине и биометрии, а также к экономике и управлению.

Большинство задач коллективного характера. Задача пред-

---

[1] Результаты приводятся на уровне значимости 0.05 .

ставляется в среднем группой из 5-6 человек, 25% задач поставлены группами из больше чем 9-10 сотрудников. Коллективным характером поставляемых задач, естественно, увеличивается их содержательная весомость.

Большинство клиентов (60%) считает предложенную им в ВЦ методику полностью подходящей, остальные - более или менее подходящей. Но вопрос, является ли примененная статистическая методика современной с точки зрения конкретной специальности на уровне республики, Советского Союза и мира утвердительный ответ дали соответственно 98%, 94% и 68% клиентов.

Ниже оценки эффективности рассматриваются более детально.

2. Информативность данных

2.1. Планировкой эмпирического исследования фиксируются измеряемые признаки, притом в значительной мере субъективно. Поэтому, формируемый для обработки массив данных может оказаться избыточным, содержать слабо относящуюся к делу часть. В целях повышения эффективности обработки естественно избегать подобных признаков, уменьшающих информативность данных в отношении к поставленной исследовательской цели. Так как исследователь часто не обладает априорными сведениями, достаточными для суждения о ценности признака, то в рассматриваемую методику включены определенные формальные показатели статистической информативности признака. Ниже рассматриваются, сложившиеся у клиента по этим показателям и по содержательной интерпретируемости результатов, оценки общей информативности обрабатываемых данных.

Самооценка информативности сравнительно высока: 60% клиентов считает свой массив вполне информативным, а 35% массивом средней информативности, так что неинформативной остается незначительная часть. Процесс обработки для выявления информативности в половине случаев протекает благоприятно, лишь ниже 10% утверждает ухудшение мнения о данных, у остальных оценка не изменялась.

Заметна слабая тенденция у женщин считать данные более часто информативными. Заодно мужчины чаще признают увеличение в ходе обработки оценки информативности.

Мера улучшения информативности положительно коррелируется с численностью рабочей группы, использующей результаты обработки (коэффициент корреляции r порядка 0.3).

Подтверждается, что более длительное консультирование с консультантами системы сопровождается с улучшением мнения о данных. Среди исследователей, имеющих консультации в объеме ниже среднего, улучшение оценки информативности встречается у 35%, а среди остальных - вдвое чаще. Вероятно, консультациями воспользовались в первую очередь те, у которых оценка информативности вначале была скромна, но тем более обоснована потребность в консультационной службе.

2.2. Состоятельную оценку можно присвоить данным лишь после их всестороннего анализа, поэтому существенно дать характеристику степени обработанности информации. В этих целях по данным анкет выводилась структура набора обрабатываемых признаков и множества полученных клиентами выпечаток, выделяя полностью, частично и слабо проанализированную информацию.

Оказывается, что в среднем пятая часть (19% ± 4%) признаков остается слабо рассмотренной, остальные считаются в равных долях основательно (41% ± 6%) и в средней мере (39% ± 5%) проанализированными. Из выпечаток примерно 15% просматриваются бегло, 30% частично и 55% в полной мере. По-видимому, имеются заметные резервы для более четкой постановки задачи, что следует учитывать и при усовершенствовании входящих в систему методов формализации содержательного задания.

Степень основательности анализа варьируется статистически значимо по количеству исследуемых в задаче признаков M. На рис. 1 приведены средние доли признаков, соответствующие разным уровням обрабатываемости, притом рассматривают три равные по количеству задач группы: 1. $M < 50$, 2. $50 \leq M < 150$, 3. $M \geq 150$. Рядом представлены аналогичные результаты о степени интерпретации выпечаток.

Рис. 1.

По этим данным можно предполагать, что количество анализируемых признаков варьируется от задачи к задаче меньше чем количество включенных в задачу признаков. Задачи второй группы, по-видимому, привлекают своими результатами сравнительно больший интерес исследователя.

Существенным фактором, влияющим на степень обработанности, можно считать величину рабочей группы (см. табл. 1). Сравнительно менее продуктивными являются группы средней численности (от 3 до 9 человек).

Таблица 1.

Средняя доля из всех выпечатков %	Величина группы		
	1-2	3-9	10 и больше
Полный анализ	57	43	71
Частичный анализ	30	40	20
Беглый анализ	12	20	8

2.3. Результативность статистической обработки в рассматриваемом классе задач в конечном виде выражается в научных публикациях. Наличие их является внешним отражением информативности данных.

Решения рассматриваемых в анкете задач нашли применение в весьма разнообразной форме: в диссертациях (результаты 70% задач), статьях (70%), докладах (60%), отчетах НИР (30%), монографиях (10%), в студенческих работах (20%) и в педагогической деятельности (30%). Большинство задач имеет реализации разного рода.

В целях большей компактности был использован сводный признак – число реализаций, пропорциональный частоте названных разных видов применения.

Выяснилась значимая положительная корреляционная связь между продолжительностью работы и числом реализаций ($r \approx 0.4$), притом не оказывались связанными между собой степень обработанности информации и численность применений. Количество признаков коррелировано с ней положительно, так же влияет больший объем консультаций. Благоприятным оказывается разнообразный план анализа – применению нескольких методов для проверки одной и той же гипотезы сопутствует сравнительно больший объем публикаций. Эти тенденции можно считать естественными, если учитывать, что в публикациях клиентов-нематематиков нередко заметная доля предусматривается для изложения соответствующей статистической методики, которая в области деятельности клиента имеет ценность оригинального научного результата.

3. Проверка рабочих гипотез исследования

3.1. Главной целью эмпирического исследования является проверка поставленных содержательных гипотез, хотя и возникновение новых, неожиданных предположений – неотъемлемая часть каждой теории. Подавляюще успешный ход проверки гипотез свидетельствует о хорошей интуиции и знании дела со стороны исследователя, но с другой стороны, указывает на корректность и эффективность применяемой службы обработки. В течение эксплуатации рассматриваемой системы статистической обработки

создавалось мнение о положительных оценках системы с такого аспекта. Анкетой это мнение подтверждается - 2/3 клиентов считает свои гипотезы доказанными полностью, остальные - в частичной мере.

Оказывается, что успешность проверки гипотез скорее связана с процессом обработки, нежели с подготовкой по анализу данных, хотя по уровню подготовки клиенты несходные. Существенными факторами успешности являются объем использованных консультаций и частота применения вспомогательных методологических пособий по системе службы. Положительно влияет большая продолжительность (значимая на уровне $\alpha = 0.01$ корреляционная связь).

Чем больше исследовательская группа, тем чаще утверждается проверяемость гипотез.

Если рассматривать те же категории по M, что и в предыдущем параграфе, то выясняется статистически существенные различия в пользу задач с $M \leq 50$ и $M \geq 150$. По-видимому, задачи с малым количеством признаков поставлены содержательно более точно, что и способствует доказательству гипотез. В задачах со многими признаками, наверное, ради обилия полученной информации, заключение о полном решении поставленной проблемы также обосновано.

Значимой является связь между объемом выборки N и возможностью проверки гипотез, отражающая большую статистическую ценность выборок большего объема. Предполагательно, зависимость успешности от N и M имеет весьма сложный характер и требует дальнейший анализ.

3.2. Опровержение или доказательство рабочих гипотез является непосредственным результатом обработки, но, заведомо, проведенный анализ имеет более широкие последствия. В данной работе были рассмотрены отзывы клиентов воздействию решенной задачи на развитие их области исследования: основоположение нового научного направления, возникновение новых гипотез, утверждение новых законов теории, постановка новых экспериментов. В табл.2 приводятся распределения полученных ответов.

Таблица 2.

%	нет	в некоторой степени	да
новые гипотезы	3	56	41
новый закон	14	46	40
новый эксперимент	9	24	67
новое направление	32	45	23

Можно считать, что исследования клиентов действуют на развитие сферы их деятельности весьма благоприятно, что также в известной мере потверждает состоятельность разработанной службы обработки.

Из соотношений показателей развития с остальными исследуемыми признаками отметим, что значимо положительно коррелированы "стаж быть клиентом" и умение преднамеревать новые эксперименты, притом общий трудовой стаж коррелируется с показателями развития, требующих большей профессиональности - выдвижение закономерностей, научных гипотез. Видимо, процесс обработки явно обогащает опыт исследователя-эмпирика.

На базе отдельных показателей развития был образован

сводный индекс развития, пропорциональный частоте утвердительных ответов у каждого компонента. Оказывается, что этот индекс коррелирован с признаками, характеризующими процесс обработки - продолжительность, воспользование методологическими пособиями и консультациями. Также, появляется значимая связь с квалификацией клиента, с его профессиональными качествами. По признакам обоих названных групп были построены линейные регрессионные модели для индекса развития и они оказались значимыми на уровне $\alpha = 0.01$.

3.3. Процесс обработки непременно оказывает действие и непосредственно на клиента, что является не менее важной стороной результативности, чем успешность развития соответствующей науки.

Абсолютно 100% опрошенных дали ответ, что проведенный анализ подтверждал их убеждение в необходимости и целесообразности применения количественных методов на их специальности. Большинство (60%) считает, что в результате обработки в значительной мере углублялись знания по анализу данных и математической статистике. Как консультации, так и методологические пособия по службе обработки благоприятно отразились на расширении знаний.

Было исследовано, как влияет служба обработки на предварительно намеченный клиентом план обработки. Оказывается, что у 40% клиентов план остается первоначальным, а у остальных - изменяется, притом у 1/5 части объем обработки сокращается.

При большем количестве консультаций изменение плана встречается более часто. Заодно с изменениями улучшается, как

правило, оценка на информативность материала. Отсюда вытекает еще один аргумент в пользу организации всесторонней службы обработки.

Отметим наконец, что женщины в процессе обработки оказываются более "резистентными" чем мужчины. Это касается как меры увеличения знаний, так и согласия на изменение плана обработки. Вдобавок, выше было указано, что и оценки информативности данных стабильнее, чем у мужчин.

4. Некоторые аспекты консультационной службы

4.1. В предыдущих разделах было приведено, что результативность обработки варьируется в зависимости от объема получаемых клиентом консультаций. Последний раздел настоящей заметки и посвящается рассмотрению этого фактора эффективности обработки.

Решение подавляющего большинства задач сопровождается консультациями, притом немаловажную роль имеют консультанты вычислительного центра, т.е. консультанты системы. В табл. 3

Таблица 3.

Объем проведенных консультаций	Консультанты ВЦ %	Консультанты вне ВЦ %
практически 0	3	58
до 3 часов	32	16
от 3 до 6 часов	23	8
от 6 до 12 часов	31	14
12 и более часов	11	4
средняя продолжительность в часах	9 ± 3	4 ± 2

приведено распределение продолжительности консультаций, откуда видно, что без консультирования обошлась незначительная часть клиентов. Продолжительности консультаций, проведенных внутри и вне ВЦ, не оказались коррелированными.

Во всех анкетах без исключения считается необходимым наличие консультантов системы.

4.2. Так как задачи обработки данных относятся к весьма разным по предмету областям исследования, то возникает вопрос о компетентности консультанта в разных задачах. Другими словами, возникает проблема, до какой степени должен и до какой степени может консультант вникнуть в содержание задачи - либо ограничиться формальными советами, либо стать по существу соавтором. Некоторое представление об ответе клиента на этот вопрос дали оценки желаемой клиентом степени вникания консультанта в задачу.

Оказывается, что глубокого проникновения в содержание задачи ждет четверть клиентов, примерно 50% считает достаточным умеренное участие консультанта в задаче, а оставшаяся пятая часть предполагает помощь консультанта, рассматривающего содержательную сторону задачи лишь в общих чертах. Итак, по этим данным консультант должен быть способен существенно варьировать вид общения с клиентами. По-видимому, неоправданным оказалась бы узкая специализация консультантов по специальностям клиентов.

Имеется зависимость от степени подготовки клиента по статистическим методам: при более слабой подготовке желательно более содержательная помощь. На рис. 2 приведена рег-

рессионная линия (с 5%–ой доверительной областью) признака, отражающего степень подготовки, в отношении к ожидаемой содержательности консультации.

Уместно отметить, что при эксплуатации рассматриваемой системы обработки регулярно проводятся семинары и курсы лекций для клиентов, что, повышая уровень подготовки клиентов, уменьшает повседневную нагрузку консультантов.

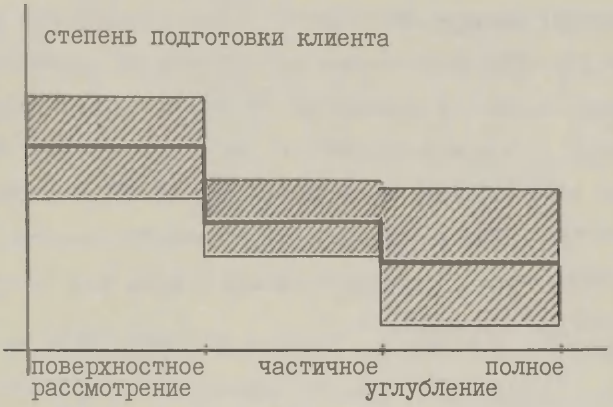


Рис. 2.

Оказывается, что женщины заинтересованы в более содержательном подходе консультанта к задаче в сравнении с мужчинами (различие при $\alpha = 0.025$). Если напомнить из предыдущего, что женщины более резистентны к влиянию процесса обработки и учитывать, что служба обработки в существенной мере действует через консультации, то можно заключить, что женщины ждут помощи в непосредственной интерпретации результатов, а мужчины исследователи – в усовершенствовании их эмпирических соображений.

4.3. Для удачного укомплектования группы консультантов службы обработки, кроме желаемой клиентом степени специализации, существенно знать, ожидается ли коллегиальный или индивидуальный образ консультации. Оказывается, что примерно половина предполагает наличие "своего" консультанта, более трети использовали бы в некоторых узловых вопросах коллегиальную помощь, а 10% вообще предпочитают консультации у нескольких консультантов.

Клиенты, имеющие опыт анализа данных и привычку работы с соответствующими методологическими пособиями, более склонны к сотрудничеству с несколькими разными консультантами. В консультантах они видят коллег, с кем целесообразно обсудить проблемы своей задачи, а не только лиц, которые проводят их через глубоко незнакомую область формального анализа.

Исследователи с большим трудовым стажем чаще предпочитают "своего" консультанта.

* * *

Что сказать в заключение?

Во-первых, следует отметить, что вследствие в основном положительных отзывов клиентов о нашей работе уменьшилась возможность изучить неблагоприятные влияния. Возникло сомнение, что у клиента не всегда хватает умений селектировать полученные результаты. Все, вышедшее "из машины", кажется одинаково ценным и мы своей методикой, хотя обязаны, не всегда можем помочь ориентироваться в статистической информации.

Во-вторых, радует, что в ходе анкетного опроса выяснилось, что оправдались наши усилия в организации службы обра-

ботки. Учитывание различий в желаниях разных категорий клиентов, вероятно, способствует более успешному решению каждой задачи.

Во-третьих, тревожит применяемая клиентами степень обработанности и интерпретации информации. Эта проблема, непосредственно связанная с затратами труда и машинного времени, требует более глубокого анализа.

Итак, если бы возникла потребность в характеристике данного исследования по применяемым выше показателям результативности, то с уверенностью можно сказать - работа являлась стимулом новых исследований, основой новых гипотез относительно управления процессом обработки.

Литература

1. Тийт Э., Общие методологические принципы системы статистической обработки данных в ВЦ ТГУ. Тр. Вычисл. центра, Тартуск. ун-т, 1977, 40, 8-13.
2. Тоодинг Л.М., Система статистической обработки данных в Вычислительном центре ТГУ. Тр. Вычисл. центра, Тартуск. ун-т, 1977, 40, 3-7.
3. Тоодинг Л.М., Описание структуры реального потока задач анализа данных. Тр. Вычисл. центра, Тартуск. ун-т, 1981, 48, 50-66.
4. Langefors, B., Analysis of user needs. Lect. Notes in Comp. Sci., 1978, 65, 1-38.

С о д е р ж а н и е

65 коп.