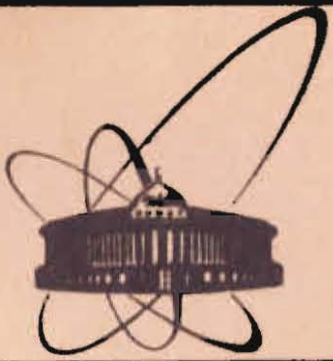

сообщения
объединенного
института
ядерных
исследований
дубна

11-85-845

А.М.Хасанов

LISS – СИСТЕМА СОПРОВОЖДЕНИЯ БИБЛИОТЕК ПРОГРАММ

Общее описание системы

1985

I. Сопровождение библиотек программ

Не новый в программировании термин "жизненный цикл программного обеспечения" отражает процесс создания и применения программ. Этот процесс состоит из нескольких стадий: определение требований к программе, проектирование, программирование, отладка и сопровождение. Каждая из них вносит свой вклад в стоимость программного обеспечения. Было проведено несколько исследований[1] величины затрат на различных стадиях жизненного цикла, и все они отмечают преобладающее значение стадии сопровождения: около 50% затрат.

Сопровождение включает внесение необходимых изменений в программы и поддержание уже готового продукта в рабочем состоянии и на уровне современных требований. Затраты на сопровождение еще выше, если программное обеспечение предназначается для широкого использования, как, например, системное математическое обеспечение или библиотеки программ общего назначения, о которых и пойдет речь.

Например, библиотека фирмы Боинг (Boeing Library) состоит из 300 программ и насчитывает свыше 24000 перфокарт. По некоторым оценкам каждая перфокарта с учетом затрат на повышение качества программ и производства широкого набора тестирующих программ обошлась в 14,5 доллара[2].

Первая версия библиотеки MSL (International Mathematical and Statistical Libraries, TEXAS) из 250 программ стоимостью в 375 тысяч долларов была объявлена для использования в 1971 году. Эта библиотека имеет высокое качество программ и непрерывно поддерживается и сопровождается фирмой. Насколько дорога деятельность по поддержанию и сопровождению библиотеки, говорит хотя бы тот факт, что через несколько лет весь проект стоил уже 1,5 миллионов долларов[2].

На производство 35 программ пакета BISPACK (проект BATS) и тестов к ним была потрачена сумма в 500 тысяч долларов. Каждая перфокарта обошлась в 20 долларов[2].

Перечисленные библиотеки и пакеты получили широкое распространение в своей стране и за рубежом. В противоположность этому фирма IBM выпустила лишь несколько версий своей библиотеки SSP [2], после чего прекратила всякое ее сопровождение. В результате эта библиотека сейчас почти не используется.

Современная библиотека программ - сложное динамическое хозяйство, в которое входят тексты программ и тестов на дисках и магнитных лентах (часто в нескольких вариантах), всевозможная документация к ним, оттранслированные программы, к которым обращается пользователь, архивные ленты, листинги прогонов программ и результатов и т.д. Трудоемкий процесс ведения этого хозяйства и составляет предмет сопровождения библиотек программ.

В него входят:

- разработка общего проекта организации библиотеки;
- комплектование библиотеки (отбор, экспертные оценки, аттестация);
- тестирование библиотеки (организация библиотеки тестов и прогон тестов);
- совершенствование библиотеки (исключение устаревших программ, замена на более эффективные, модификация программ);
- пополнение библиотеки (добавление новых программ);
- трансляция и перетрансляция библиотеки и отдельных программ;
- исправление ошибок в программах (которые обнаруживаются в процессе активной работы с ними);
- архивная работа (периодический сброс библиотек на магнитные ленты и восстановление после порчи библиотеки и отдельных программ);
- перевод программ на повышенную точность;
- ведение документации (создание и модификация описаний, оперативные исправления, тиражирование);
- адаптация библиотеки на ЭВМ других типов;
- работа с библиотекой в связи с изменениями в операционной системе, появлением новых трансляторов;
- консультация пользователей;
- передача библиотеки в другие организации.

При таком объеме работ снижения затрат на сопровождение и увеличения производительности труда сопровождающих программистов и качества библиотек можно достичь только развитием автоматизированных средств сопровождения. Что обычно применяет сопровождающий программист ? Трансляторы, ассемблеры, редакторы связей, редакторы текстов, системные сервисные программы. В его работе были бы полезны и следующие средства:

- компараторы (сравнивают два текста, например, варианты программы, результаты тестирования);
- верификаторы (проверяют программу на соответствие некоторому стандарту языка);
- конверторы (автоматизируют перевод программ с одного языка программирования на другой. Отметим, что при сопровождении библиотеки "Дубна" применяется достаточно хороший конвертор /3/);

- преобразователи форматов (форматоры: приводят текст программы к удобочитаемому виду, см. программу Q900 (TIDY) в библиотеке "Дубна" /4,8/);
- системы управления версиями (например, средствами системы PATCHY /7/);
- системы, выдающие структурную схему программ (см. программу Q800 (FLODIA) /4,8/);
- развитый аппарат процедур (по переписи, распечатке и т.п.);
- сборщик оперативных данных во время работы программы (частота использования операторов, максимальные и минимальные значения и т.д.);
- интерактивные отладчики;
- системы составления пакетов заданий;
- системы ведения журнала ошибок и изменений;
- системы аварийной выдачи;
- системы получения статистической информации по использованию библиотек и отдельных программ.

В данной работе описывается первый вариант универсальной системы ведения небольших банков данных и ее применение для сопровождения библиотек программ - система LISS (LIbrary Support System).

2. Общее описание системы

Система LISS была создана для сопровождения библиотеки "Дубна". Тем не менее она носит общий характер и может применяться при сопровождении любых других библиотек и больших программных комплексов.

Система предназначается для ведения банков данных, содержащих сведения о каких-либо объектах внешнего мира. Например, о библиотеках, программах, книгах и т.д. Информация представляется с помощью атрибутов (признаков) объекта: год издания, число программ, цвет и т.п. Каждый атрибут имеет свое имя. Атрибуты имеют значения, которые будем называть элементами данных, например, 1980, зеленый. В системе предусмотрено три типа элементов данных и, соответственно, три формы их кодирования:

NAME - строка символов определенной длины. Символы: цифры и буквы латинского алфавита;
INT - целые неотрицательные числа;
SET - множества элементов из дискретного набора.

Соответственно, атрибуты могут быть трех типов: NAME , INT , SET .

Например, атрибут INDEX может иметь значение B200, атрибут YEAR - 1985, а атрибут COLOUR может иметь значение, являющееся множеством: { GREEN, RED } . Множество может состоять из одного или нескольких элементов, может быть и пустым. INDEX имеет тип NAME , YEAR - тип INT , COLOUR - тип SET .

Набор атрибутов, относящийся к объекту, называется записью. Будем называть записью также и соответствующий набор значений этих атрибутов, находящийся в банке. Запись может состоять из любого количества атрибутов каждого типа, но в любой записи должен быть хотя бы один атрибут типа NAME .

Объекты могут быть двух видов (уровней): главного и подчиненного (или исходного и порожденного). Может существовать также несколько типов объектов подчиненного уровня. Эти объекты существуют только в связи с объектами главного уровня.

Каждый объект в банке данных представляется своей записью. Записи объектов главного уровня отличаются от записей объектов подчиненного уровня,так же,как последние различаются между собой. Таким образом, в банке существует столько типов записей, сколько есть типов объектов подчиненного уровня плюс тип записей объектов главного уровня. Каждый тип записи имеет свое имя. Отметим, что записей подчиненного уровня может и не быть, тогда имеется только один тип записей - для главного уровня.

Записи главного и подчиненного уровней связаны между собой в виде двухуровневого дерева, или в терминологии рабочей группы по базам данных ассоциации CODASYL /5/, в виде набора. В набор входит один экземпляр записи главного уровня и любое количество экземпляров каждого типа записей подчиненного уровня. Узлами дерева являются записи соответствующих объектов:

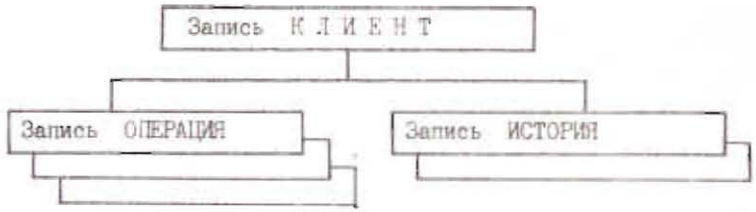

Рис.I. Двухуровневое дерево записей, или набор.

Банк данных представляет собой иерархический файл из некоторого количества таких наборов:

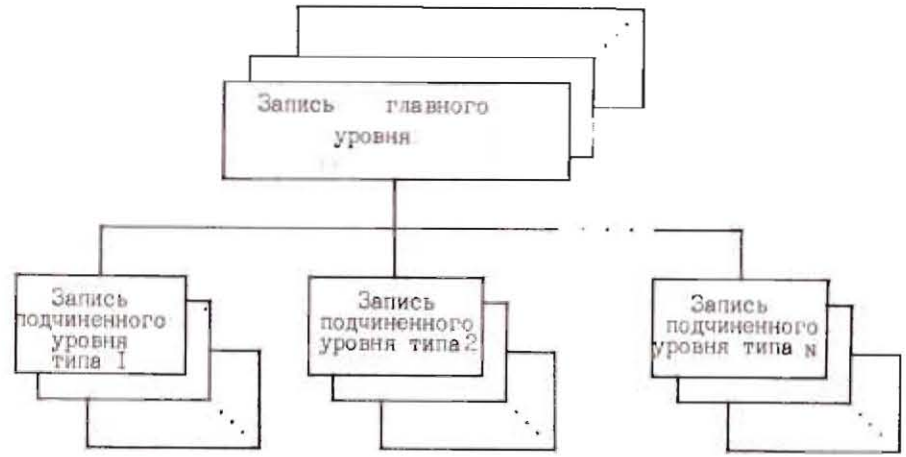

Рис.2. Схема банка данных.

Определим область как подраздел банка данных. Каждая область имеет свое имя. Каждый набор входит в одну и только одну область. Объединение всех областей составляет банк данных. Возможно любое количество областей, но не более числа наборов в банке данных. Если разделения на области нет, то весь банк данных составляет одну область.

Мы описали логическую структуру, схему банка данных. На основе этой схемы можно составить множество подсхем в зависимости от того, для каких целей будет применяться данная система. Подсхема - это описание конкретного банка данных, с которым будет работать система. В системе существует специальный простой язык для описания подсхем. В следующем разделе приведен пример подсхемы для системы сопровождения библиотеки "Дубна". Описание подсхемы помещается в файл INP . В этом же файле находятся описания объектов банка данных. Для их записи также используется некоторый язык, соответствующий языку описания подсхем.

Чтобы задать подсхему, надо определить, сколько типов записей будет в банке данных, для каждого типа указать его имя, а также структуру записи, т.е. описать, какие атрибуты входят в данный тип записи и в каком количестве. Для каждого атрибута надо указать его имя и тип,к которому относится атрибут (NAME , INT или SET). Для атрибутов типа SET , кроме того, надо задать список всех его возможных значений. Атрибут типа NAME , заданный первым, называется ключом типа записи, т.е. он идентифицирует запись. Ключ записи главного типа является ключом набора. Если для записей подчиненных уровней уникальность ключа не требуется, то ключ набора должен определять набор однозначно.

Чтобы настроить систему на конкретную подсхему, надо сделать в ней небольшое количество изменений, фактически изменить значения некоторых констант. Это константы MAXSTR, INTSIZEST , определяющие количество символов и, соответственно, цифр в данных типа NAME и INT , константа MAXSERV , определяющая количество символов в именах типов, атрибутов и в данных типа SET . Это константы MAXN , MAXI , MAXT , определяющие максимальное количество атрибутов типов NAME , INT и SET в одной записи, и MAXD – максимальное количество значений атрибутов типа SET в одной записи. Эти константы можно задавать с некоторым превышением, чтобы не перетранслировать программу, если количество атрибутов или значений атрибутов типа SET увеличится. Надо задать также константу CVARSYST , которая определяет имя системы. Например, для системы сопровождения библиотек задана константа CVARSYST='LISS /I.8 ДУБНА'- Library Support System, вариант I.8.

Каждый раз при запуске системы происходит инициализация банка данных. Она проходит в два этапа.

На первом система настраивается на подсхему, описанную в файле INP , то есть заполняет таблицы имен типов записей, атрибутов, их значений, проверяет непротиворечивость подсхемы. Если на этом этапе обнаруживается ошибка, то система выдает сообщение о ней и заканчивает работу.

На втором этапе происходит собственно заполнение банка данных в соответствии с подсхемой. При этом данные считываются из файла INP , анализируются и записываются (в память ЭВМ) специальным образом, обеспечивающим минимизацию занимаемой памяти, быстрый доступ и переработку. Одновременно производится синтаксический и семантический контроль данных. В случае ошибок выдаются сообщения, указывается место и причина ошибки.

3. Система сопровождения библиотеки "Дубна"

Покажем, как описанная выше система была применена для сопровождения библиотеки программ общего назначения "Дубна". Система LISS вместе с банком данных представляет собой информационно-административную систему, которая автоматизирует процесс сопровождения библиотеки. В банке данных содержится документация по сопровождению библиотеки, но в сжатом виде. Вся информация о программах, которую можно выразить кратко: числом, строкой или другой единицей, внесена в банк данных.

Итак, система LISS предоставляет программисту, сопровождающему библиотеку, следующие возможности:

-содержать банк данных в соответствии с историей развития библиотеки;

- оперативно получать разнообразную полную и достоверную информацию по современному состоянию библиотеки и по ее развитию;
- выводить различные документы и справки по библиотеке в форматированном виде;
- сравнивать реальную библиотеку (оттранслированных программ на диске) с информацией о ней, заложенной в банке данных.

Заметим, что последняя возможность - это расширение системы, описанной в предыдущем разделе, для нужд сопровождения библиотеки. Это, так сказать, ее библиотечно-зависимая часть, которая не потребуется при других применениях системы. Так как автор занимается, в основном, вопросами библиотеки, то в дальнейшем эта часть системы, по-видимому, будет расширяться. Более детально о возможностях системы будет сказано ниже.

Программы, входящие в библиотеку, обычно состоят из подпрограмм и подпрограмм-функций. Будем называть их одним словом подпрограмма. Программа может содержать одну программу, может иметь сложную структуру, например, быть пакетом. Программы могут обращаться к подпрограммам, ей не принадлежащим. Это внешние подпрограммы. Таким образом, с каждой программой связан некоторый набор подпрограмм и их имен. Некоторые подпрограммы имеют также дополнительные (ENTRY) имена.

В системе LISS объектами главного уровня являются программы, а объектами подчиненного - подпрограммы. Существует три типа объектов подчиненного уровня: подпрограммы, имена которых являются именами для пользователя, подпрограммы, имена которых являются внутренними, и подпрограммы, имена которых являются внешними именами программы.

Таким образом, в банке данных есть записи четырех типов. Имя типа записей главного уровня - PROGRM , имена типов записей подчиненного уровня соответственно: USERN , INTERN , EXTERN.

В настоящее время записи в банке данных имеют следующую структуру (заметим, что эта структура не является жестко закрепленной, а может изменяться):

PROGRM - записи этого типа содержат общие сведения о программах библиотеки.

Атрибуты типа NAME :

INDEX - индекс программы (например, A100, C200 /B/);
REPLCE - индекс адекватной программы или программы, которой можно пользоваться взамен устаревшей или исключенной;
COMMON - имена общих блоков программы.

Атрибуты типа INT :

USNUM - количество подпрограмм типа USERN в программе;

INTNUM - количество подпрограмм типа INTERN ;

EXTNUM - количество подпрограмм типа EXTERN ;

COMSIZ - размеры общих блоков;

ACCRCY - точность счета (количество точных десятичных цифр результата);

SPACE - оперативная память, требуемая для счета;

PERFO - длина программы;

Атрибуты типа SET (в скобках перечислены имена значений, которые может принимать атрибут):

AUTHOR (CERN,JINR,LIHP)
- организация, из которой получена программа;

COMPUT (EC1060,CDC,BESM6)
- на ЭВМ какого типа внедрена программа в ОИЯИ;

DATA (DOUBLE,SIMPLE,INTEGR,REAL,COMPLX,LOGICL,CHAR)
- типы данных, с которыми оперирует программа;

LANG (FORTRN,ASSEMB)
- язык программирования, на котором написана программа;

STATUS (SUPPRT,ELIMIN,OBSOLT)
- статус программы; здесь сокращения от слов SUPPORT , ELIMINATE, OBSOLETE - программа поддерживается, исключена из библиотеки, устарела;

STRUCT (SUBR,FUNCT,MAIN,PACKGE)
- структура программы;

WRITUP (SHORT,LONG)
- описание программы, краткое или подробное. Здесь сокращение от WRITEUP - подробно описывать;

REFILE (INPUT,OUTPUT,PUNCH,USFILE,TAPE,NOFILE)
- файлы, с которыми может работать программа: INPUT - чтение, OUTPUT - печать, PUNCH - перфорация, USFILE - файлы пользователя, TAPE - файлы в ФОРТРАНе для CDC , NOFILE - файлов нет.

Заметим, что атрибуты типа SET могут принимать не обязательно одно, а сразу несколько значений (или множество) из перечисленных. Программа может быть внедрена на нескольких машинах, тогда атрибут COMPUT имеет, например, значение {EC1060 , CDC} . Если в программе есть чтение с перфокарт, печать результатов и обращение к файлу пользователя, то REFILE ={INPUT,OUTPUT,USFILE} .

Атрибут INDEX типа NAME является ключом записи типа PROGRM , а также ключом набора.

USERN/INTERN - записи этих типов имеют одинаковую структуру и содержат сведения о конкретных подпрограммах пакета.

Атрибуты типа NAME :

NAME - имя подпрограммы для пользователя или внутренней подпрограммы:

Атрибуты типа SET :

COMPUT (EC1060,CDC,BESM6)
- иногда одна программа имеет на разных машинах разные имена подпрограмм или дополнительные имена;

LANG (FORTRN,ASSEMB)
- язык, на котором написана данная подпрограмма. Аналогично, подпрограммы с одинаковыми именами могут быть написаны на разных языках на разных (или даже на одной) машинах;

STATUS (SUPPRT,ELIMIN,OBSOLT)
- статус подпрограммы. Подпрограмма на одной машине может использоваться, а на другой быть исключенной;

ENTRY (MODULE,ENTRY)
- тип точки входа в подпрограмму. Имя может быть именем основного входа - MODULE, или дополнительного - ENTRY;

TYPROC (SUBR,FUNCT,MAIN)
- тип программного модуля: подпрограмма SUBROUTINE , подпрограмма-функция FUNCTION , головной модуль MAIN ;

TYPE (USERN,INTERN)
- тип записи USERN или INTERN , к которому относится данное имя.

Атрибутов типа INT нет.

EXTERN - эти записи содержат сведения о внешних ссылках данной программы.

Атрибуты типа NAME :

NAME - имя внешней ссылки;

INDEX - индекс внешней подпрограммы, если она находится в этой же библиотеке (C300, D500), имя USER , если это подпрограмма пользователя, имя SYSTEM , если это системная программа.

Атрибуты типа SET :

TYPE (EXTERN)
- тип записи.

Атрибутов типа INT нет.

Покажем, как все это записать на языке описания подсхем. Для записи используются позиции 1-71. Если информация не помещается на строке, то возможен перенос на любое количество дополнительных строк. Запись на строке должна заканчиваться либо запятой, либо закрывающей скобкой, после чего через пробел может идти комментарий. Строка продолжения начинается пробелом, запись начинается со второй позиции.

```
TO PROGRM  начало описания записей главного уровня
NAME       описание атрибутов типа  NAME
3(INDEX,   индекс программы
 REFLCE,   индекс адекватной программы
 COMMON)   имена общих блоков
INT        описание атрибутов типа  INT
7(USERNM,INTERNM,EXTERNM,COMSIZ,ACCESS,SPACE,PERPO)
SET        описание атрибутов типа  SET
8(AUTHOR,COMPUT,DATA,LANG,STATUS,STRUCT,WRITUP,REFILE)
1 3(CERN,JINR,LINP) значения атрибута AUTHOR
2 3(EC1060,CDC,BESM6) значения атрибута COMPUT
3 7(DOUBLE,SIMPLE,INTEGR,REAL,COMPLX,LOGICL,CHAR)
4 2(FORTRN,ASSEMB)  язык программы
5 3(SUPPRT,ELIMIN,OBSOLT)
6 4(SUBR,FUNCT,MAIN,PACKGE)
7 2(SHORT,LONG)
8 5(INPUT,OUTPUT,PUNCH,USFILE,TAPE)
STANDSET(1,1-2-3,4,1,1,1,1,*)
T1 USERN  описание записей типа  USERN
NAME
1(NAME)   имя подпрограммы для пользователя
SET
6(COMPUT,LANG,STATUS,ENTRY,TYPROC,TYPE)
1 3(EC1060,CDC,BESM6)
2 2(FORTRN,ASSEMB)
3 3(SUPPRT,ELIMIN,OBSOLT)
4 2(MODULE,ENTRY)
5 3(SUBR,FUNCT,MAIN)
6 2(USERN,INTERN)
STANDSET(1-2-3,1,1,1,1,1)
T2 INTERN описание записей типа  INTERN  совпадает с описанием
          записей типа  USERN
```



```
STANDSET(1-2-3,1,1,1,1,2)
T3 EXTERN  описание записей типа  EXTERN
NAME
2(NAME,INDEX)
SET
1(LEVEL)
1 1(EXTERN)
STANDSET(1)
```

Строка STANDSET устанавливает стандартные значения атрибутов типа SET . Каждое число задает номер значения для соответствующего атрибута. Если номера перечислены через дефис (черточку), то значением атрибута является множество. Например, стандартное значение атрибута COMPUT в записях типа PROGRM есть множество из трех элементов: {EC1060, CDC, BESM6} , стандартное значение атрибута STATUS - SUPPRT . Символ * означает, что стандартное значение соответствующего атрибута не определено, или, по-другому, стандартным является неопределенное значение.

Как видно, описания можно задавать с комментариями (так описаны атрибуты типа NAME в записях типа PROGRM) или без них (так описаны атрибуты типа INT).

Описание подсхемы помещается, как уже было сказано, в файл INP .

Для окончательной настройки системы на данную подсхему, зададим в ней следующие значения констант:

```
MAXSTR    = 6
INTSIZEST = 6
MAXSERV   = 6
MAXN      =10
MAXI      =10
MAXT      =10
MAXD      =30
CVARSYST  =' LISS/1.8 ДУБНА  '
```

Здесь константы заданы с некоторым превышением: для описываемого варианта максимальное количество атрибутов типа NAME -три, типа INT - семь, а не десять.

В этом же файле INP , после описания подсхемы, помещается описание объектов банка данных, в данном случае программ и подпрограмм.

Приведем для примера описание конкретной программы и ее подпрограмм:

```
0 D(D114,,ANSWER-INTERN-OPTION-PARAMS-  описание записи
 RANDOM-STORAG),(,,,,,,2,2),(1,2,1,     главного типа
 2-8-3-4-2-1064)
1(RIWIAD)      экземпляр записи типа USERN
2(RIWIBD),(1) четыре экземпляра
2(RANDU),(1)  записей типа INTERN
2(RIWIA),(3)
2(STD114),(3)
3(RNDM,V104)  два экземпляра
3(EXAMPL,USER)записей типа EXTERN
```

Номер в первой позиции показывает, к какому типу относится запись. Запись каждого типа может состоять их трех частей, заключенных в скобки. В первой паре скобок задаются значения атрибутов типа NAME , во второй - атрибутов типа SET , и в третьей - типа INT .

Если атрибут имеет несколько значений, они перечисляются через дефис. Если атрибут не имеет значения или это значение пока не известно, то вместо него ставится запятая. Запятые перед закрывающей скобкой можно опустить.

Так как для атрибутов типа SET в описании подсхем задан стандартный набор значений, то значения, совпадающие со стандартными, также можно опускать. Если весь набор значений совпадает со стандартным, то эту часть описания можно опустить полностью, как, например, в описании подпрограмм RIWIAD, RNDM и EXAMPL . В описаниях остальных подпрограмм со стандартным набором не совпадает по одному значению. Например, подпрограммы RIWIBD и RANDU есть только на ЭВМ ЕС-1060, а подпрограммы RIWIA и STD114 - только на БЭСМ-6.

Иногда значения некоторых атрибутов не определены, например, они неизвестны на данный момент времени, или мы сомневаемся в их достоверности. Тогда, как было уже сказано, для атрибутов типа NAME и INT эти значения просто опускаются, а для атрибутов типа SET на это место необходимо поставить символ ж, так как иначе вместо них будет принято стандартное значение.

Если в описании подсхемы запись не имеет атрибутов какого-либо типа (NAME,INT или SET), то соответствующая часть также отсутствует в описании объекта. Вместо отсутствующих частей ставится запятая, если есть продолжение описания.

Буква D в описании записи Ø-го типа указывает, к какой области банка данных относится объект. В нашей системе имена и структура областей совпадают с именами и структурой разделов библиотеки "Дубна". Таким образом, в банке данных есть области A,B,C,... Z .

Имя области не обязательно должно состоять из одной буквы, оно может быть строкой символов, то есть относится к типу данных NAME .

4. Допустимые изменения в системе

Любое программное обеспечение со временем развивается. Нельзя представлять и банк данных,как нечто застывшее и неизменное. Он может расширяться, реорганизовываться, может изменяться подсхема, структура записей, могут быть добавлены новые типы записей, новые атрибуты, изменяться их имена. Данная система создавалась с учетом того, чтобы как можно реже перетранслировать программу при изменениях в банке данных.

Приведем примеры возможных изменений с указанием , какие преобразования они вызывают.

1. Использование системы для нового приложения. Например, мы использовали ее для создания системы сопровождения библиотек, а теперь применим для создания каталога прочитанных книг и статей. При этом должна быть изменена программа (изменение констант) и файл INP (новое описание подсхемы и описания объектов банка данных). Словом, должно быть сделано все то, что описано в разделе 3.

2. Использование старого приложения к новым условиям. Например, система сопровождения конкретной библиотеки применяется для сопровождения другой библиотеки. При этом подсхема банка данных может остаться прежней. Программу также не обязательно изменять и перетранслировать. Все изменения касаются только второй части файла INP с описаниями объектов банка данных.

Следующие изменения рассматриваются в рамках одного приложения.

3. Развитие банка данных. Добавляются описания новых объектов, изменяются и уничтожаются старые записи. Изменяется только вторая часть файла INP .

4. Изменение подсхемы: новый тип записи. Например, первоначально в системе сопровождения библиотек использовалось только два типа записей: PROGRM - главного уровня и SUBPRG - подчиненного. В записях типа SUBPRG содержались сведения о всех подпрограммах;и для пользователя и внутренних. Указание на имена внешних ссылок содержалось в записях PROGRM . Затем было решено, что целесообразнее выделить эти сведения в отдельные записи. Так,вместо записей типа SUBPRG появились три новых типа: USERN , INTERN и EXTERN .

Так как к тому времени банк данных еще не был создан, то изменения коснулись только описания подсхемы. Заметим, что добавление нового типа записей при уже созданном банке данных - довольно сложное дело, и лучше все такие изменения проводить на этапе проектирования банка данных.

5. Изменение структуры типа записи: новые атрибуты. Например, в системе сопровождения библиотек в типе записей EXTERN были атрибуты только типа NAME . Затем был добавлен атрибут типа SET для указания, к какому типу записи относится подпрограмма.

В типе записей PROGRM будет, по-видимому, добавлен атрибут MODIFY типа INT для указания на даты модификации программ. Так как любой атрибут может иметь несколько значений, то,задавая даты модификации через дефис,мы будем иметь историю развития программы. Каждый раз при изменении программы мы добавляем в банк данных дату этой последней модификации.

Такие изменения (добавление новых атрибутов) можно производить последовательно. Сначала мы изменяем только описание подсхемы, а затем постепенно и описания объектов банка данных. Важно в подсхеме имена новых атрибутов ставить в конец списка атрибутов. Тогда, до тех пор, пока мы не внесем изменения в описания объектов в банке данных, значения этих атрибутов будут считаться неизвестными, а для атрибутов типа SET стандартными.

6. Изменение списка значений атрибутов типа SET . Например, так как в ОИЯИ появилась новая ЭВМ ЕС-1061, то имя EC1061 будет добавлено в список значений атрибута COMPUT .

В перечень значений атрибута TYPROC в типах USERN и INTERN следует добавить новое значение BLDATA для программных единиц типа BLOCK DATA.

Раньше в списке значений атрибута REFILE стояло только 5 значений (без NOFILE). Из этих значений трудно было выбрать стандартное, поэтому в строке подсхемы, описывающей стандартные значения атрибутов типа SET , на соответствующем месте стоял символ *: STANDSET (1,1-2-3,4,1,1,1,1,*), обозначая этим, что стандартное значение не определено. Теперь за стандартное принято новое значение NOFILE , и в STANDSET на этом месте стоит 6.

Изменения этого типа производятся обычно довольно легко и касаются только описания подсхемы и описания тех объектов, у которых значение нового атрибута не совпадает со стандартным.

Заметим, что, если значения максимальных констант в программе были заданы с некоторым превышением, и произведенные изменения укладываются в их границы, то предыдущие три типа изменений не требуют модификации и перетрансляции программы.

7. Новое имя типа, атрибута, значения.
Изменяется имя типа записи, имя атрибута или имя значения атрибутов типа SET .

Эти изменения обычно производятся для того, чтобы добиться наиболее точного мнемонического обозначения. Например, имя LOGICL

заменило имя BOOL в списке значений атрибута DATA , так как оно больше соответствует типу данных LOGICAL в программах на языке FORTRAN .

Такие изменения производятся только в описании подсхемы и не требуют дополнительных действий.

8. Новая форма кодирования элементов данных и имен. Изменяется форма представления данных, а также имен типов записей, атрибутов и значений атрибутов типа SET . Например, изменить максимальное количество символов в данных типа NAME , количество цифр в данных типа INT : вместо шестизначных номеров телефонов вводятся восьмизначные. Или изменить количество символов в именах типов, атрибутов и их значений. Вместо FORTRN - FORTRAN , вместо PROGRM - PROGRAM , вместо ENTRY - ENTRYPOINT.

Если изменение идет в сторону увеличения, то они производятся только в программе, не затрагивая банка данных.

Уменьшать количество символов в уже работающей системе не рекомендуется.

9. Новый тип данных. Добавить к трем типам данных NAME , INT и SET новый тип данных, например, FLOAT.

Изменения такого типа в системе невозможны.

ЛИТЕРАТУРА

I. Гласс Р., Нуазо Р. Сопровождение программного обеспечения, "Мир", М., 1983.
2. Вестник Московского университета. Сер. Вычислительная математика и кибернетика, I, 1977.
3. Хасанов А.М. ОИЯИ,11-83-824, Дубна, 1983.
4. Федорова Р.Н., Хасанов А.М. ОИЯИ, 11-82-674, Дубна, 1982.
5. Мартин Дж. Организация баз данных в вычислительных системах, "Мир", М., 1978.
6. Лукстиня Л. и др.,ОИЯИ, P11-85-170, Дубна, 1985.
7. Klein H.,Zoll J. PATCHY-4, CERN, 1977.
8. CERN Computer Centre, Program Library -I,II.
9. Current trends in programming methodology, Prentice-hall, Inc., Englewood Cliffs, New Jersey, 1977.

Рукопись поступила в издательский отдел
25 ноября 1985 года.

НЕТ ЛИ ПРОБЕЛОВ В ВАШЕЙ БИБЛИОТЕКЕ?

Вы можете получить по почте перечисленные ниже книги, если они не были заказаны ранее.

	Труды УШ Всесоюзного совещания по ускорителям заряженных частиц. Протвино, 1982 /2 тома/	11 р. 40 к.
Д17-81-758	Труды II Международного симпозиума по избранным проблемам статистической механики. Дубна, 1981.	5 р. 40 к.
Р18-82-117	Труды IV совещания по использованию новых ядерно-физических методов для решения научно-технических и народнохозяйственных задач. Дубна, 1981.	3 р. 80 к.
Д2-82-568	Труды совещания по исследованиям в области релятивистской ядерной физики. Дубна, 1982.	1 р. 75 к.
Д9-82-664	Труды совещания по коллективным методам ускорения. Дубна, 1982.	3 р. 30 к.
Д3,4-82-704	Труды IV Международной школы по нейтронной физике. Дубна, 1982.	5 р. 00 к.
Д11-83-511	Труды совещания по системам и методам аналитических вычислений на ЭВМ и их применению в теоретической физике. Дубна, 1982.	2 р. 50 к.
Д7-83-644	Труды Международной школы-семинара по физике тяжелых ионов. Алушта, 1983.	6 р. 55 к.
Д2,13-83-689	Труды рабочего совещания по проблемам излучения и детектирования гравитационных волн. Дубна, 1983.	2 р. 00 к.
Д13-84-63	Труды XI Международного симпозиума по ядерной электронике. Братислава, Чехословакия, 1983.	4 р. 50 к.
Д2-84-366	Труды 7 Международного совещания по проблемам квантовой теории поля. Алушта, 1984.	4 р. 30 к.
Д1,2-84-599	Труды VII Международного семинара по проблемам физики высоких энергий. Дубна, 1984.	5 р. 50 к.
Д17-84-850	Труды III Международного симпозиума по избранным проблемам статистической механики. Дубна,1984. /2 тома/	7 р. 75 к.
Д10,11-84-818	Труды V Международного совещания по проблемам математического моделирования, программированию и математическим методам решения физических задач. Дубна, 1983	3 р. 50 к.
	Труды IX Всесоюзного совещания по ускорителям заряженных частиц. Дубна, 1984 /2 тома/	13 р.50 к.

Заказы на упомянутые книги могут быть направлены по адресу:
101000 Москва, Главпочтамт, п/я 79
Издательский отдел Объединенного института ядерных исследований

Хасанов А.М. 11-85-845

LISS - система сопровождения библиотек программ.
Общее описание системы

Дано общее описание первого варианта универсальной системы ведения небольших банков данных и системы /Library Support System/ - ее применения для сопровождения библиотек программ. Описывается также логическая структура /схема/ банка данных, приводится язык задания подсхем. Банк данных системы LISS содержит в сжатом виде всевозможные характеристики библиотеки и отдельных ее программ /автор, язык, ЭВМ, структура и др./. Там же находится инструкция по использованию системы.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1985

Перевод О.С.Виноградовой

Khasanov A.M. 11-85-845

LISS - A Library Support System. General Description of the System

General description of the first variant of small universal data base management system and of library support system (LISS) are given. The data base logical structure is described the language for description of subschemes is given. The LISS data base has the condensed characteristics of the library and its programs (author, language, computer, structure etc..) The system guide is presented.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1985