«ФИЗИКА ЭЛЕМЕНТАРНЫХ ЧАСТИЦ И АТОМНОГО ЯДРА»,
1975, том 6, вып. 3.

УДК 681.142

ПРИНЦИПЫ ОРГАНИЗАЦИИ И СТРУКТУРА МОДУЛЬНОЙ СИСТЕМЫ ПРОГРАММ ОБРАБОТКИ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

Н. Н. Говорун, Л. Дорж,
В. Г. Иванов,
А. Ф. Лукьянцев

Объединенный институт ядерных исследований, Дубна

В работе рассматриваются принципы организации и структура модульной системы HYDRA, предназначенной для анализа экспериментальных данных. Система состоит из набора служебных подпрограмм, предназначенных для расширения возможностей основного алгоритмического языка системы (ФОРТРАН), подпрограмм общего назначения и наборов модулей. Обмен данными между модулями прикладных программ производится только через блок динамически распределяемой памяти с помощью специальных подпрограмм системы.

Principles of organization and the structure of HYDRA modular system intended for experimental data analysis are considered. The system consists of a set of service subroutines to extend resources of the basic algorithmic language (FORTRAN) of the system, common subroutines and processors. Data exchange between applied programmes is only performed through a dynamic storage by means of special subroutines.

ВВЕДЕНИЕ

Пузырьковые и искровые камеры широко используются для исследования процессов, происходящих при взаимодействии частиц высоких энергий с веществом. С помощью этих установок ежегодно получают миллионы стереофотографий с различного рода случаями ядерных взаимодействий.

Для анализа этого потока экспериментальных данных создаются и развиваются системы обработки фильмовой информации, состоящие из сложной просмотрово-измерительной аппаратуры, электронно-вычислительных машин и комплекса разнообразных программ [1]. Математическое обеспечение систем обработки фильмовой информации состоит из программ контроля и управления работой просмотрово-измерительных устройств, программ фильтрации и предварительной обработки результатов измерений,

а также программ для реконструкции пространственной картины событий, их кинематического и статистического анализа.

Наиболее широко для обсчета результатов обмера камерных фотографий используется цепочка программ THRESH — GRIND — AUTGGR — SLICE — SUMX [2]. Эти программы разработаны в ЦЕРНе и их различные модификации используются в настоящее время в большинстве ядерных центров. Программы написаны в основном на алгоритмическом языке ФОРТРАН и рассчитаны на широкий класс различных трековых установок.

Чтобы облегчить работы по развитию и хранению разнообразных версий программ обработки, последние необходимо разделить на отдельные секции, из которых специальная программа PATCHY собирает и готовит требуемый для трансляции вариант. Внедрение секционной структуры позволило значительно ускорить работы по модернизации программ обработки и упорядочить систему их хранения.

В последние годы в практике программирования успешно развивается тенденция создания больших программ обработки экспериментальных данных из отдельных модулей [3]. На этой базе созданы геометрические и кинематические программы для больших пузырьковых камер [4] и камер классического типа, программа статистического анализа и ряд других программ обработки экспериментальных данных.

Принципы организации модульной системы программ обработки экспериментальных данных (система HYDRA) были разработаны в ЦЕРНе [3]. Основным алгоритмическим языком системы является ФОРТРАН, возможности которого расширены за счет специальных системных подпрограмм, предназначенных для решения следующих задач:

организации динамически распределяемой памяти и работы с ней;

передачи управления от одного модуля к другому;

регистрации различного рода ситуаций, встречающихся в процессе обсчета данных и передачи управления от текущего модуля модулю более высокого уровня в случае необходимости;

распечатки динамически распределяемой памяти или ее частей.

организации ввода — вывода данных;

построения гистограмм и графиков;

работы с файлами;

ввода и записи в динамически распределяемую память блоков информации (титлов).

Обмен данными между отдельными модулями системы, называемыми также процессорами, производится только через динамически распределяемую память, в которой каждый модуль находит исходные данные и засылает в нее результаты своих вычисле-

ний. Такая организация связи между отдельными элементами системы обеспечивает реальную независимость отдельных модулей.

Система HYDRA состоит из ряда системных подпрограмм, наборов модулей и подпрограмм общего назначения. Конкретные программы в рамках этой системы составляются из ее элементов, вызываемых из управляющей программы в необходимой последовательности. Система HYDRA в короткий срок завоевала широкое признание, и работы по ее адаптации для различных ЭВМ ведутся в большом числе ядерных центров Европы. В частности, работы по постановке этой системы на ЭВМ БЭСМ — 6 [5] ведутся в Объединенном институте ядерных исследований и Институте физики высоких энергий АН ГДР. Работа по постановке системы HYDRA на ЕС ЭВМ ведется также в ЦИФИ (ВНР), Физическом институте АН ЧССР, Институте экспериментальной физики САН (ЧССР) и некоторых других институтах.

Внедрение модульных принципов в практику программирования и перевод программ обработки экспериментальных данных на модульную структуру позволяют не только сократить время, затрачиваемое на создание математического обеспечения физических экспериментов, но и значительно упростить процедуру обмена программами между различными организациями.

Переход на модульную систему вместо дальнейшего развития классической цепочки программ [2] был также обусловлен следующими обстоятельствами:

1) работы по созданию программ обработки ведутся большими коллективами, состоящими из сотрудников различных институтов. Чтобы успешно провести такие работы, необходимо создавать программы, которые пригодны для широкого класса ЭВМ, имеющих трансляторы с языка ФОРТРАН;

2) строительство новых больших пузырьковых камер с очень сложной оптикой, с неперекрывающимися полями зрения и т. п. потребовало создания новых геометрических программ. В ходе этих работ были разработаны и реализованы более совершенные принципы организации больших программ;

3) значительное увеличение статистики в камерных экспериментах потребовало для сокращения затрат машинного времени адаптации универсальных программ обработки данных к требованиям конкретных экспериментов.

Все это привело к созданию модульной системы, которая со временем заменит классическую цепочку программ обработки фильмовой информации.

Данный обзор посвящен рассмотрению основных принципов организации модульной системы HYDRA, предназначенной в основном для обработки фильмовой информации, и ее структуры [2,4], знакомит читателя с основными идеями системы, организацией модулей, программ и основными элементами системы.

1. ОБЩАЯ ОРГАНИЗАЦИЯ СИСТЕМЫ

В качестве основного алгоритмического языка, на котором составляются элементы системы (модули, системные и вспомогательные подпрограммы) выбран ФОРТРАН, или, более точно, его версия ANSI FORTRAN. Этот язык хорошо известен экспериментаторам, так как на нем создана обширная библиотека разнообразных программ для обработки данных и проведения научно-технических расчетов. Кроме того, трансляторами с алгоритмического языка ФОРТРАН снабжены основные ЭВМ, имеющиеся в различных ядерных центрах.

На автокоде написано всего лишь несколько подпрограмм, предназначенных для работы с битами, байтами и символами, а также для организации передачи управления в тех случаях, когда это нельзя сделать на ФОРТРАН. Кроме того, на автокоде также составляются наиболее часто используемые подпрограммы, фортрановские варианты которых требуют заметных затрат машинного времени.

Так как ФОРТРАН не обладает всеми требуемыми системой возможностями (динамическое распределение памяти, составление переменных указателей к спискам структур данных и т. п.), то для этих целей в системе имеются специальные подпрограммы, которые расширяют возможности основного алгоритмического языка.

Служебные подпрограммы системы объединены в несколько групп (пакетов) и предназначены для решения следующих задач:

организации и работы с динамически распределяемой памятью;

передачи управления от одного модуля системы к другому;

записи в динамическую память блоков информации и работы с ними;

подсчета различных ситуаций, встречающихся в процессе обсчета данных и передачи управления модулям более высокого уровня в тех случаях, когда текущие модули не могут «решить вопрос» о дальнейшей судьбе анализируемых данных. Например, переполнение арифметического устройства, деление на нуль и т. п. ситуации;

выдачи на печать содержимого динамической памяти или ее участков;

организации ввода — вывода данных;

построения гистограмм и графиков;

работы с файлами;

генерации искусственных событий.

В рамках модульной системы программы составляются с помощью управляющих подпрограмм из отдельных модулей. Каждый модуль состоит из одной или ряда фортрановских подпрограмм и предназначен для решения четко определенной задачи с заданной ему структурой данных.

В отличие от обычных программ обработки, которые являются для большинства пользователей «черным ящиком», каждый модуль решает строго определенную конкретную задачу, например вычисление коэффициентов преобразования от одной координатной системы к другой по заданным координатам реперных точек в этих двух системах. При этом точно известны форма задания исходных для обсчета данных и форма записи результатов. Ввиду того что все модули системы имеют краткие, но достаточно четкие описания, составление из них требуемых программ не представляет особых затруднений для подготовленного пользователя.

Таким образом, модульная система HYDRA состоит из набора модулей, предназначенных для решения конкретных задач обработки данных, системных подпрограмм, расширяющих возможности основного алгоритмического языка, и небольшой библиотеки подпрограмм общего назначения.

2. ОРГАНИЗАЦИЯ ДИНАМИЧЕСКИ РАСПРЕДЕЛЯЕМОЙ ПАМЯТИ В ПРОГРАММАХ МОДУЛЬНОЙ СТРУКТУРЫ

Общая организация динамической памяти. Динамически распределяемая память программы — это непомеченный общий блок, длина которого задается в главной программе. Любая подпрограмма может использовать этот блок, для этого требуется включить описывающие его операторы COMMON, DIMENSION и EQUIVALENCE в декларативные операторы соответствующей подпрограммы.

В динамической памяти хранятся все данные, необходимые для работы программы, за исключением локальных данных подпрограмм, локальных переменных системных подпрограмм, а также информации, хранящейся в помеченных общих блоках системы.

Информация, записываемая в динамическую память, делится на отдельные группы (банки), которые организуют определенные структуры данных. В процессе работы программы ее динамическая память делится системой на три основные части:

Данные прямого доступа	Резервная память	Данные непрямого доступа

Данные прямого доступа (ДПД) занимают нижнюю часть блока. Эта часть памяти в свою очередь делится на две части: ДПД системы и ДПД программы пользователя. В данных прямого доступа системы указываются адреса банков (связи) и структур данных, требуемых для работы системных подпрограмм, которые находятся среди данных непрямого доступа. Эта часть памяти заполняется и используется только системными подпрограммами. Параметры,

характеризующие распределение динамической памяти, хранятся в специальном помеченном блоке системы.

ДПД программы предназначена для хранения адресов начальных банков структур данных последней части блока, а также используется в качестве рабочей памяти модулей программы. При переходе от одного модуля к другому содержимое рабочей памяти полностью уничтожается. Поэтому вся информация, которая необходима для работы других модулей, должна быть сформирована в виде соответствующих банков и заслана в динамическую память. Таким образом, часть памяти, отведенная для данных прямого доступа состоит из трех частей: связей (адресов банков) системы, связей программы и рабочей памяти, используемой для операций с данными.

Данные непрямого доступа (или место накопления банков) содержат информацию, распределенную по банкам переменной длины, которые объединены в определенные структуры данных. Адреса начальных банков структур задаются в данных прямого доступа системы или программы. Банки создаются с помощью специальных системных подпрограмм и располагаются в блоке, начиная с его конца (верхний адрес). После создания нового банка его адрес передается вызывающей подпрограмме для записи в данные прямого доступа или засылки в другой банк, структурно связанный с только что созданным. Конкретный адрес слова данных в банке определяется через адрес банка и положение этого слова в банке.

Резервная память — это свободная часть памяти между данными прямого доступа и местом накопления данных. Она также используется в качестве рабочей памяти нескольких системных подпрограмм, и ее длина не может быть меньше заданного минимума (100 ячеек). Увеличение числа банков или рабочей памяти приводит к уменьшению резервной памяти. Когда длина резервной памяти достигает заданного минимального значения, то вызывается специальная системная подпрограмма, которая удаляет из памяти все уже не нужные банки данных, а оставшиеся сдвигает в верхнюю часть блока, ликвидируя зазоры между ними.

Рассмотрим организацию динамической памяти на следующем простом примере:

```
DIMENSIØN Q (999), IQ (999)
EQUIVALENCE (Q, IQ, LQUSER)
CØMMØN / / LQUSER (7), LQMAIN, LQSYS (30)—ДПД системы
+,          LINKS (24), LB, LK (3)—связи программы
+,          FIRSTD, A (9, 9), B, C, N  рабочая память
+,          LASTD                      рабочая память
```

ДПД программы

Динамической памятью в данном примере является вектор Q, начало которого совпадает с началом данных прямого доступа.

Пусть LA — адрес банка A, тогда значение i-го слова банка определяется следующим выражением:

$$PI = Q(LA + i).$$

Чистка памяти. В процессе работы программы может оказаться, что содержащаяся в ряде банков информация больше не нужна программе и ее можно выбросить. В этом случае вызывается специальная подпрограмма, которая в одном из разрядов статусного слова банка устанавливает флаг (специальную метку), указывающий на то, что банк больше не нужен программе. Отмеченные таким образом банки логически исключаются из соответствующих структур данных, но остаются в памяти до тех пор, пока она полностью не заполнится. Когда в памяти нет места для создания новых банков или расширения рабочей памяти, вызывается специальная системная подпрограмма, которая удаляет из памяти ненужные банки, а все оставшиеся сдвигает в верхнюю часть таким образом, чтобы вся неиспользуемая программой резервная память располагалась одним куском. При этом также производится переопределение адресов всех сдвигаемых банков.

Чистка памяти и перемещение банков требуют разделения используемых программой адресов на постоянные и переменные (или переопределяемые). К последним относятся адреса связанных между собой банков, структур данных, которые изменяются в процессе чистки памяти и фиксируют начала соответствующих банков. Относительные адреса же слов, данных внутри банков, остаются неизменными, а их абсолютные адреса определяются адресом банка и порядковым номером слова в данном банке. Поскольку практически любое обращение к системе может вызвать перераспределение памяти, то адреса начальных банков структур хранятся среди данных прямого доступа, а адреса связанных с ними банков — в специальных ячейках, называемых связями. Такая организация облегчает и ускоряет процесс поиска нужных адресов.

При работе с динамически распределяемой памятью необходимо тщательно следить за правильной и своевременной засылкой адресов банков в нужные места, так как «ошибочные связи» могут приводить к весьма неожиданным результатам.

Системные подпрограммы для работы с динамически распределяемой памятью. Системные подпрограммы для работы с динамически распределяемой памятью объединены в М-пакет (Memory-package). Подпрограммы этого пакета предназначены для определения размеров общего блока, выделения требуемого участка рабочей памяти, формирования банков данных, преобразования данных из рабочей памяти в банки, чистки памяти и т. п. Рассмотрим назначение основных подпрограмм этого пакета.

Подпрограмма MQINIT (LAST) предназначена для засылки и присвоения начальных значений соответствующим параметрам

подпрограмм М-пакета, создания блока динамически распределяемой памяти и задания его длины. Она вызывается в начале работы программы раньше всех остальных системных подпрограмм. Параметр LAST — последнее слово блока динамически распределяемой памяти. Например,

```
+CDE, Z=Q
      +, SPACE (4000), LAST
```

Подпрограмма MQWORK (FIRSTD, LASTD) используется в программах для выделения требуемого участка рабочей памяти. Здесь FIRSTD — первое, а LASTD — последнее слово рабочей памяти. Например:

```
+CDE, Z=Q
      +, LINKS (24), LB, K (3)
      +, FIRSTD, A (9, 9) B, C, N, LASTD
```

Рабочую память необходимо затребовать до ее использования.

Подпрограмма MQLIFT (LS, K, N, NAME) предназначена для создания в динамической памяти банка с параметрами, задаваемыми в массиве NAME, LS — адрес этого банка. Значения параметров K и N определяются структурой данных, в которую включается создаваемый банк.

Подпрограмма MQGARB производит чистку памяти и переопределение адресов банков, перемещаемых в ходе этого процесса.

Подпрограммы М-пакета обеспечивают эффективное и достаточно простое использование динамически распределяемой памяти программы.

3. СТРУКТУРЫ ДАННЫХ

Информация, хранимая в динамической памяти программы, состоит из отдельных групп данных, которые называются банками. Под банком в системе понимается группа данных, которая состоит из следующей информации: идентификатора, адресов банков, связанных с данными (связи банка), текущего состояния банка, характеризуемого специальными флагами, и собственно данных.

Идентификатор банка содержит соответствующим образом упакованные следующие данные: четырехсимвольное название банка в А-формате (ID) и три числа NL, NS и ND, определяющие соответственно число связей, число структурных связей и число данных.

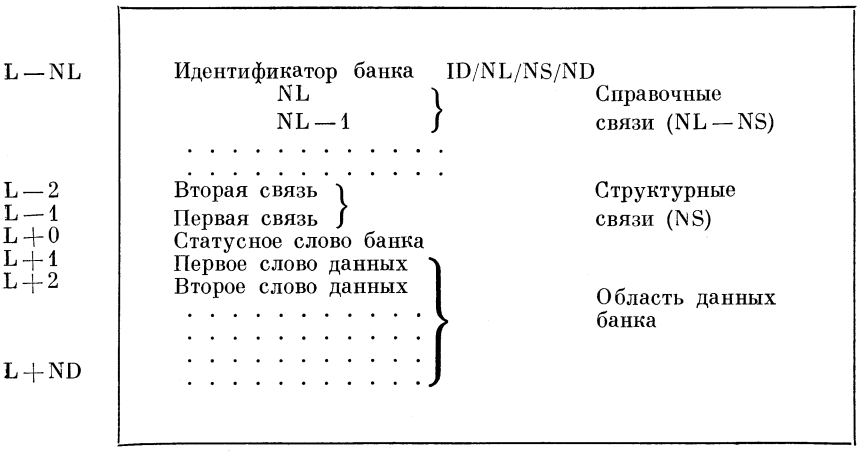
Связи банка — это целые положительные числа, определяющие адреса банков, связанных с данным, в динамически распределяемой памяти. Общее число связей банка (NL) не должно быть больше 511, а число структурных связей (NS) — 63. Связи банков переопределяются в процессе чистки памяти.

Каждый банк может иметь связи двух типов: справочные и структурные. Справочная связь двух банков A и B означает, что данные банка B можно использовать при анализе данных, содержащихся в банке A. Адрес справочной связи может указывать любое слово банка B, включая его адрес. Если содержимое банка B уже не требуется программе и система исключает его из динамической памяти, то одновременно соответствующее значение связи полагается равным нулю. Структурные связи банков предназначены для образования в памяти структур связанных между собой данных. Адреса структурных связей в процессе чистки памяти или полагаются равными нулю, или переопределяются. Более подробно назначение структурных связей будет рассмотрено позднее.

Информация о текущем состоянии банка содержится в специальном слове, называемом статусным словом банка. Адрес статусного слова банка является одновременно и адресом банка. Разряды статусного слова банка содержат одноразрядные флаги, предназначенные для указания системе действий с данной группой информации. Пятнадцать младших разрядов статусного слова отведены для флагов программы. Номера разрядов, отводимых для тех или иных флагов, задаются в специальном общем блоке системы. К ним относятся метка выброса банка, метка системного банка и т. п.

Область данных банка содержит числовой материал. Обычно это вещественные числа, представленные в форме чисел с плавающей запятой. Максимальное число слов, которые можно отвести под данные, определяются размерами оперативной памяти ЭВМ, отведенной для программ пользователей.

Информация внутри банка располагается в следующем порядке:



Адресом банка является адрес его статусного слова, в данном случае L. Все данные банка находятся в его ячейках, номера которых больше L, а связи — в ячейках, номера которых меньше L. Причем, если IS — номер структурной связи банка, то эта связь находится в ячейке банка с номером (L—IS).

Название банка можно определить с помощью системной подпрограммы QNAME, использование которой требует относительно больших затрат машинного времени. В связи с этим рекомендуется помещать название банка среди данных в тех случаях, когда он часто используется в программе, скажем, в первое или любое другое фиксированное слово данных.

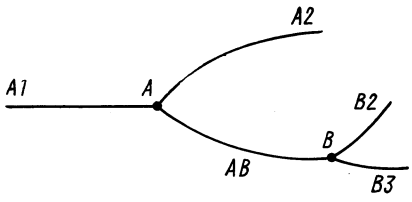


Рис. 1

Организация структур данных. Система организации банков и задание в них адресов других банков позволяет строить в динамической памяти структуры данных. Любой банк, находящийся в динамической памяти, принадлежит к одной из структур данных. Обычно каждый банк структуры связан с другим, в котором содержится его адрес. Исключения составляют лишь начальные банки структур, адреса которых находятся среди данных прямого доступа системы.

Если адрес банка B (LB) находится в списке связей банка A, то это означает их логическую связь. Такая система позволяет выражать логические связи между различными группами данных в простом и общем виде.

В системе HYDRA имеется два типа структурных связей: вертикальные и горизонтальные. Рассмотрим эти связи на примере события, изображенного на рис. 1. Налетающая частица (A1) образует двулучевое взаимодействие в точке A (треки A2 и AB). Одна из вторичных частиц (AB) образует в точке B вторичное двулучевое взаимодействие (треки B2 и B3). Результаты обмера этого события вместе с необходимой для анализа служебной информацией можно представить в виде следующей структуры данных (рис. 2).

Банк события организует всю структуру данных и его адрес находится среди данных прямого доступа. Из рис. 2 видно, что элементы структуры имеют вертикальные и горизонтальные связи. Вертикальные связи соединяют элементы структуры, находящиеся между собой в определенной подчиненной последовательности. Например, событие и его первая вершина, вершины события и выходящие из них треки. Горизонтальные связи соединяют между собой элементы структуры одного типа. Например, вершины события или треки, выходящие из данной вершины. По принятому в систе-

ме соглашению первая связь банка структуры данных является горизонтальной, а вторая — вертикальной. Так, в нашем примере вторая связь банка события должна указывать адрес банка его первой вершины. Первая связь банка вершины А — адрес банка

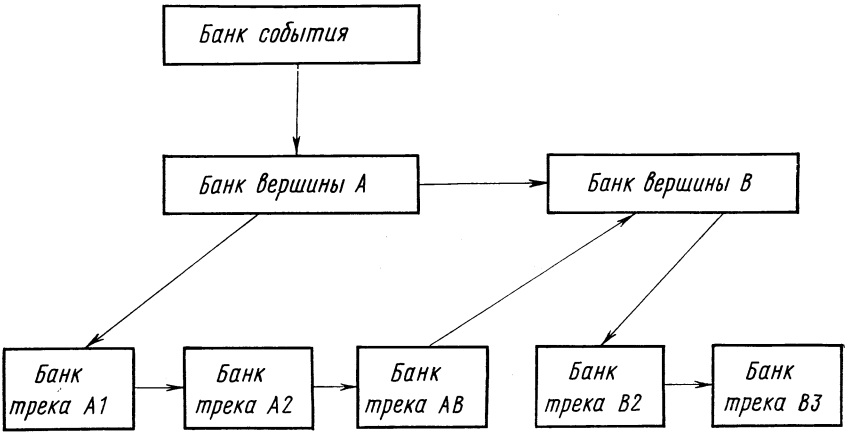


Рис. 2

вершины В, а вторая — адрес банка трека А1. Первая связь банка трека А1 — адрес банка трека А2, а вторая связь банка трека А2 в данном случае должна равняться нулю и т. д. Трек АВ соединяет две вершины события А и В, и поэтому его банк связан справочной связью с банком вершины В. Рассмотренную структуру данных можно схематически показать следующим образом (рис. 3).

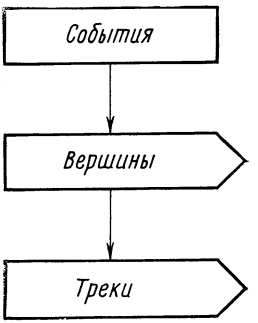
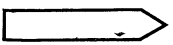


Рис. 3

Событие, его вершины и треки соединяются вертикальными связями, вершины события и треки, выходящие из одной вершины,— горизонтальными. Последние обозначаются символом  . Все банки, связанные между собой горизонтальными связями, называются линейной структурой. Банки одного и того же типа связываются между собой горизонтально по принятому в системе условию только через их первые связи. Если в банке имеется несколько структурных связей, то первая связь резервируется для указания адреса следующего по порядку банка такого же типа. Нулевой адрес означает, что в структуре нет больше банков такого типа. Если же в последнем банке линейной структуры ука-

зан адрес ее первого банка, то такая структура называется циклической линейной. В большинстве случаев порядок расположения банков линейной структуры не имеет значения.

В вертикальной структуре порядок следования банков имеет определенное значение, и его нельзя менять произвольно. Рассмотрим это обстоятельство на примере события, изображенного на рис. 1. Предположим также, что для определения параметров треков

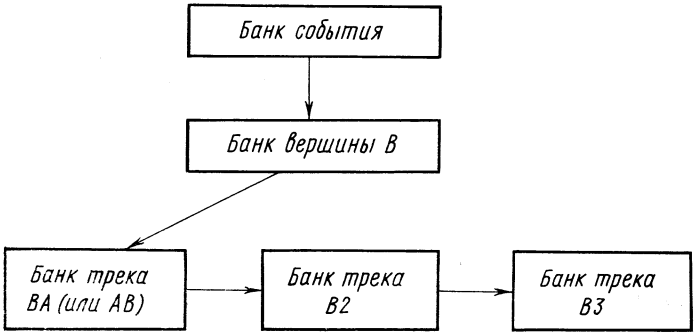


Рис. 4

необходимо точно знать координаты вершины А. Тогда в случае плохого измерения координат этой вершины программа не смогла точно определить ее координаты и банк вершины А помечен как не нужный. В этом случае нет необходимости заниматься реконструкцией треков, выходящих из этой вершины, а в большинстве случаев и всего события. Иногда можно ограничиться только выбросом вершины А и связанных с ней треков, за исключением АВ, так как последний связан также и с вершиной В. В этом случае в структуре данных остаются только вторая вершина и связанные с ней треки, а сама структура принимает вид, приведенный на рис. 4.

Система позволяет производить выброс ненужных банков из структур данных анализом и прослеживанием их связей, с помощью специальных подпрограмм, но это требует заметных затрат машинного времени. Поэтому в системе имеются специальные подпрограммы, которые позволяют маркировать ненужные банки в программе пользователя, не проводя анализа их связей.

Маркировка структур данных. При чистке памяти струтурные связи банков переопределяются, так как часть банков структуры можно выбросить, а оставшиеся сдвинуть в верхнюю часть памяти. При этом адреса уже ненужных банков структуры заменяются на адреса следующих за ними оставшихся банков. Маркиров-

ка конкретных структур производится с помощью специальных системных подпрограмм QTØUCH и QTCHID.

При обращении к подпрограмме QTØUCH в качестве параметров задается номер разряда статусного слова, в который необходимо занести соответствующий флаг (0 или 1), адрес начального банка, организующего структуру данных, и цепочка BCD символов, определяющих условия маркировки структуры. Подпрограмма маркирует все банки структуры данных организуемыми вертикальными и горизонтальными связями начального банка. В качестве условий используются следующие символы: S, H, Z (S означает, что необходимо маркировать начальный банк структуры; H требует маркировки структуры данных, организуемой первой связью банка, т. е. горизонтальной связью. Если буква H не задана в числе условий, то подпрограмма отмечает только те банки структуры, которые связаны с начальными вертикальными связями; Z требует записи нулей в заданный разряд статусных слов банков структуры).

Рассмотрим работу этой подпрограммы на следующем примере (рис. 5):

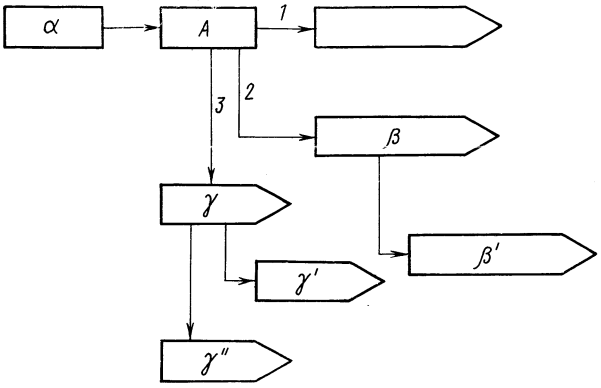


Рис. 5

Банк A, входящий в линейную структуру α, организует две сложные подструктуры β и γ с помощью второй и третьей связей. Номера связей показаны на рис. 4. Для того чтобы маркировать все банки этих подструктур, необходимо обратиться к подпрограмме со следующими значениями параметров:

CALL QTØUCH (IBIT, LA, 0).

Здесь IBIT — номер разряда в статусных словах маркируемых банков; LA — адрес банка A.

Для того чтобы маркировать только банки подструктуры γ, нужно вызвать подпрограмму QTØUCH со следующими параметрами:

CALL QTØUCH (IBIT, IQ (LA-3), 2HH).

Подпрограмма QTCHID идентична подпрограмме QTØUCH в логике следования по цепочкам связей структуры, но с ее помощью можно маркировать (или не маркировать) банки с заданными названиями. Названия отмечаемых (или не отмечаемых) банков задаются в качестве фактических параметров при обращении к подпрограмме. Так как подпрограммы QTØUCH и QTCHID требуют относительно больших затрат машинного времени на прослеживание связей структуры или поиск заданных банков, их не рекомендуется часто использовать. Наиболее экономично, с точки зрения машинного времени,— использование системной подпрограммы QDRØP.

Подпрограмма QDRØP (К, JL) предназначена для установления в статусном слове банка флага выброса и логического исключения его из структуры данных. Иначе говоря, помеченный подпрограммой банк остается в памяти, но и уже не входит в структуру данных. При чистке памяти он будет удален из нее уже физически. При обращении к этой подпрограмме задаются два параметра:

К-адрес маркируемого банка структуры, т. е. адрес слова в динамической памяти, в котором находится структурная связь этого банка. Так, если адрес банка А, структурно связанного с банком S, находится в четвертой ячейке связей банка, то К-адрес банка А КА = LS — 4;

JL -флаг, определяющий работу подпрограммы. Если JL = 0, то это значит, что флаг выброса устанавливается только для заданного банка. Если JL = 1, то программа устанавливает флаг выброса для всех банков структуры, связанных с начальным его первой связью.

4. МОДУЛИ СИСТЕМЫ И ОРГАНИЗАЦИЯ СВЯЗИ МЕЖДУ НИМИ

Основные элементы, из которых составляются программы системы — модули. С формальной точки зрения, они отличаются от обычных фортрановских подпрограмм только способом передачи управления. Если в обычных фортрановских подпрограммах передача управления производится с помощью операторов CALL и RETURN, то в модулях это делается с помощью специальных системных подпрограмм. Обмен данными между модулями программы производится только через банки данных в динамически распределяемой памяти, адреса которых задаются вызываемому модулю в специально создаваемом для него банке вызова.

Модули могут вызываться как из главной программы, так и из любых других модулей. Каждый модуль решает строго определенную конкретную задачу с заданными ему данными и может состоять из одной или нескольких подпрограмм, число которых с точки зрения системы не ограничено.

Рассмотрим теперь организацию программ в рамках модульной системы и основные требования к ее модулям.

1. Структура и организация конкретной программы определяются главной программой, составляемой пользователем в соответствии с требованиями и задачами эксперимента.

2. Программы составляются из набора имеющихся в системе модулей.

3. Каждый модуль может состоять из одной главной и любого числа вторичных подпрограмм и предназначен для решения четко определенной конкретной задачи.

4. В модуле имеется только один вход, которым является вход в его главную подпрограмму.

5. Исходные для модуля данные (входная структура данных) готовятся другими и находятся в банках динамически распределяемой памяти.

6. Для каждого вызываемого модуля создается специальный банк вызова, в котором задаются адреса банков входной структуры данных, а также другие параметры, необходимые для его работы и работы системных подпрограмм. Адрес банка вызова запоминается в системной части данных прямого доступа.

7. Локальные переменные, используемые внутри данного модуля, не сохраняются при переходе к другому, если они не занесены в соответствующие структуры данных динамической памяти.

8. Любой модуль может использовать всю рабочую память массива данных прямого доступа. В связи с этим при вызове нового модуля рабочая память полностью очищается.

9. Для сохранения данных, локальных для данного модуля, в системе имеется возможность создания банков промежуточных результатов. В отличие от других банков системы банки этого типа уничтожаются при выходе из данного модуля. Создание банков промежуточных результатов необходимо в тех случаях, когда модуль вызывает несколько других и при этом требуется сохранить результаты проводимых им вычислений, которые хранятся в ячейках рабочей памяти.

10. Результаты вычислений, проведенных данным модулем, образуют выходную структуру данных и также заносятся в динамически распределяемую память.

Таким образом, модули системы по существу являются отдельными группами фортрановских подпрограмм, на которые наложен ряд ограничений. Эти ограничения в основном касаются только

главных подпрограмм модулей и не распространяются на вызываемые ими подпрограммы.

Назначение, структура и образование банков вызова. Рассмотрим более подробно структуру, назначение и формирование в программах банков вызова.

Банк вызова создается и запоминается в динамически распределяемой памяти перед обращением к каждому модулю. Название банка вызова совпадает с названием вызываемого модуля. Через банк вызова передаются адреса банков входной структуры и соответствующие параметры. Он создается и тогда, когда не используется вызываемым модулем. В этом случае в нем запоминается адрес возврата.

Идентификатор банка
Адрес верхнего банка вызова
Адрес банка промежуточных данных
Связи вызывающего модуля
Связи вызываемого модуля
Статусное слово банка
Параметрические данные для вызываемого модуля
Данные для вызывающего модуля
Системные параметры или адрес возврата, параметры рабочей памяти. (Эта часть данных зависит от характеристик ЭВМ)

Рис. 6

Для каждого модуля существуют банки вызова двух типов: создаваемый для данного модуля вызывающим его и создаваемый данным модулем при вызове им нового. Банки вызова первого типа называются верхними, второго — нижними банками вызова. Специальная системная подпрограмма объединяет их в структуру банков вызова (рис. 6).

Доступ к информации, хранящейся в банках вызова, производится через их адреса, которые запоминаются в нижней части динамической памяти:

LQUP — адрес верхнего банка вызова текущего модуля;

LQDW — адрес нижнего банка вызова. Перед обращением к модулю он равен нулю;

LQAN — адрес предпоследнего нижнего банка вызова;

LQSV — адрес банка промежуточных данных.

Чтобы создать банк вызова и включить его в соответствующую структуру данных, необходимо включить в подпрограмму следующие операторы ФОРТРАН:

```
DIMENSIØN MPRØC (4)
DATA MPRØC (4HPRØC, NL, 0,  ND)
. . . . . .
. . . . . .
CALL JQBØØK (MPRØC)
. . .  . . . . . .
```

В этом случае системная подпрограмма JQBØØK создает в динамической памяти программы банк вызова с названием, заданным в MPRØC(1), числом связей — NL и слов с данными — ND.

Иногда при переходе к новому модулю требуется сохранить содержимое отдельных частей рабочей памяти. Для этой цели создается специальный банк хранения промежуточных результатов. Создание этого банка производится включением в соответствующую подпрограмму следующей группы операторов:

```
DIMENSIØN MSX (4)
DATA MSX (2HSX, NL, NS, ND)
. . . . . .
. . . . . .
CALL MQLIFT (LQSV, 0, 0, MSX)
. . . . . .
```

Здесь LQSV — адрес созданного банка промежуточных результатов; MSX(1) — название банка; MSX(2) — число связей банка; MSX(3) — число структурных связей банка; MSX(4) — число данных банка.

Все нужные для создания банка данные задаются в модуле перед обращением к системной подпрограмме MQLIFT.

Передача управления от одного модуля к другому. Передача управления от одного модуля к другому производится в следующей последовательности:

1) создание банка вызова;

2) создание банка для хранения промежуточных результатов, если это требуется;

3) передача управления от вызывающего модуля вызываемому с помощью специальной системной подпрограммы JQJUMP;

4) передача управления от вызываемого модуля вызывающему, которая производится с помощью специальной системной подпрограммы JQBACK.

Передача управления вызываемому модулю производится с помощью системной подпрограммы JQJUMP. Как уже отмечалось, каждый модуль имеет только один вход — это вход в его главную подпрограмму. Положение точки входа в модуль передается подпрограммам системы через оператор EXTERNAL. При передаче управления вызываемому модулю адрес его верхнего банка вызова присваивается переменной LQUP, а адрес нижнего банка вызова LQDW полагается равным нулю. Вызов модуля реализуется с помощью следующих операторов ФОРТРАН:

```
EXTERNAL PRØC
. . . . . .
. . . . . .
CALL JQBØØK (MPRØC)
. . . . . .
CALL JQJUMP (PRØC)
. . . . . .
```

Возврат в вызывающий модуль производится с помощью системной подпрограммы JQBACK. При этом из динамической памяти исключаются все нижние банки (если таковые имеются), вызывавшегося модуля, а границы рабочей памяти и значения адресов банков вызова и промежуточных результатов (LQUP, LØDW, LQSV) восстанавливаются до значений, которые они имели перед вызовом.

В связи с тем что обращение к модулю выглядит несколько сложнее, чем обращение к обычной подпрограмме, рассмотрим процедуру вызова на нескольких простых примерах.

Пример 1. Модуль A1 вызывает модуль B1. Сделаем два предположения:

1) начальный адрес структуры входных данных модуля A1 задается первой связью его банка вызова;

2) начальный адрес структуры входных данных модуля B1 задается первой связью, а выходной структуры данных — второй связью его банка вызова.

В этом случае вызов модуля B1 будет выглядеть следующим образом:

```
      SUBRØUTINE A1
+CDE, Z=Q
      +, LNEW
       . . . . . .
      EXTERNAL B1
      DIMENSIØN MB1 (4)
      DATA MB1 (2HB1, 2, 0, 0)
       . . . . . .
       . . . . . .
      CALL JQBØØK (MB1)          }  Создание банка вызова и вызов
      IQ (LQDW-1)=IQ (LQUP-1)    }  модуля B1.
      CALL JQJUMP (B1)           }
      LNEW=IQ (LQDW-2)           Запоминание начального адреса выходной структуры данных
       . . . . . .               B1 для дальнейшего использования
       . . . . . .
      CALL JQBACK
      END
```

Пример 2. Последовательный вызов двух модулей B1 и B2.

В дополнение к предположениям первого примера предположим также, что модуль B2 использует в качестве входной структуры данных (первая связь его банка вызова) выходную структуру данных модуля B1 и присваивает адрес своей выходной структуры данных второй связи банка вызова.

В этом случае последовательное обращение к двум модулям будет выглядеть следующим образом:

```
      SUBRØUTINE A1
+CDE, Z=Q
      +, LIN, FIRSTD, X (20), LASTD
      EXTERNAL B1, B2
      DIMENSIØN MB1 (4), MB2 (4)
      DATA MB1 (2HB1, 2, 0, 0)
      DATA MB2 (2HB2, 2, 0, 0)
       . . . . . .
       . . . . . .
      CALL MQWØRK (FIRSTD, LASTD)
      LIN=IQ (LQUP-1)
       . . . . . .
       . . . . . .
       . . . . . .
      CALL JQBØØK (MB1)
      IQ (LQDW-1)=LIN
      CALL JQJUMP (B1)

      LIN=IQ (LQDW-1)
       . . . . . .
       . . . . . .
       . . . . . .
       . . . . . .
       . . . . . .
       . . . . . .
      CALL JQBØØK (MB2)
      IQ (LQDW-1)=IQ (LQAN-2)
      CALL JQJUMP (B2)
       . . . . . .
       . . . . . .
       . . . . . .
      CALL JQBACK
      END
```

вызов модуля B1

После возвращения из B1 рабочая память восстанавливается, но ее содержимое потеряно, поэтому чтобы использовать LIN, необходимо восстановить ее значение

вызов модуля B2

Напомним, что LQAN определяет адрес предпоследнего нижнего банка вызова.

Пример 3. Последовательный вызов трех модулей (B1, B2, B3) с созданием банка для хранения промежуточных результатов.

В дополнение к предположениям предыдущих примеров предположим, что модуль B3 использует в качестве входной структуры данных (первая связь банка вызова) выходную структуру модуля B2 и оставляет ее без изменений и, кроме того, структура выходных данных B1 должна быть использована в дальнейшем и ее следует сохранить при вызове B3.

```
      SUBRØUTINE A1
+CDE, Z=Q
      +, LIN, LØUTB1, FIRSTD, X (20), LASTD
      . . . . . .
      EXTERNAL B1, B2, B3
      DIMENSIØN MB1 (4), MB2 (4), MB3 (4), MSAVE (4)
      DATA MB1 (2HB1, 2, 0, 0)
      LATA MB2 (2HB2, 2, 0, 0)
      DATA MB3 (2HB3, 1, 0, 0)
      DATA MSAVE (4HSAVE, 1, 0, 7)
      . . . . . .
      CALL MQWØRK (FIRSTD, LASTD)
      LIN=IQ (LQUP-1)                  } Выделение рабочей памяти и создание банка промежуточных результатов
      CALL MQLIFT (LQSV, 0, 0, MSAVE)  }
      . . . . . .
      . . . . . .
      Q (LQSV+5)=X (9)                 Содержимое X (9) занесено в ячейку банка промежуточных результатов для дальнейшего использования.

      CALL JQBØØK (MB1)
      IQ (LQDW-1)=LIN                  } Вызов B1
      CALL JQJUMP (B1)                 }
      LØUTB1=IQ (LQDW-2)
      IQ (LQSV-1)=LØUTB1               Запоминание адреса выхода B1
      . . . . . .
      . . . . . .

      CALL JQBØØK (MB2)
      IQ (LQDW-1)=LØUTB1               } Вызов B2
      CALL JQJUMP (B2)                 }
      . . . . . .
      . . . . . .
      CALL JQBØØK (MB3)
      IQ (LQDW-1)=IQ (LQAN-2)          } Вызов B3
      CALL JQJUMP (B3)                 }
      X (9)=Q (LQSV+5)
      LØUTB1=IQ (LQSV-1)               Восстановление значений X (9) и LØUTB1
      . . . . . .
      . . . . . .
      CALL JQBACK
      END
```

Системные подпрограммы J-пакета. Передача управления от одного модуля к другому и выполнение необходимых для этой цели подготовительных операций производится системными под-

программами, объединенными в J-пакет (Jump-package). Назначение трех подпрограмм этого пакета JQBØØK, JQJUMP и JQBACK было рассмотрено в данном разделе.

Задание первоначальных значений параметров, используемых подпрограммами этого пакета, и проведение необходимых для их работы подготовительных операций производятся подпрограммой JQINIT, которую следует вызывать после вызова системной подпрограммы MQINIT, если в программе имеются два или большее число модулей.

В связи с тем что не все нужные передачи управления можно реализовать на ФОРТРАН, в подпрограммы JQJUMP и JQBACK включены небольшие подпрограммы, написанные на автокоде. Такая комбинация ФОРТРАН и автокода может работать на большинстве ЭВМ. Если это оказывается невозможным, то подпрограммы перехода следует писать только на автокоде.

Необходимые для работы подпрограммы J-пакета параметры системы и программы хранятся в специальных помеченных блоках системы. К ним относятся: число системных связей, добавляемых к каждому банку вызова; число системных данных, добавляемых к каждому банку вызова (параметры рабочей памяти, адрес возврата и другие данные, значения которых следует сохранять); название верхнего банка вызова модуля, который вызвал текущий; название верхнего банка вызова текущего модуля; название нижнего банка вызова текущего модуля; номер уровня текущего модуля (номер уровня управляющей программы равен нулю) и т. п.

5. ДИАГНОСТИЧЕСКИЕ ВОЗМОЖНОСТИ СИСТЕМЫ

При отладке программ на электронно-вычислительных машинах хорошая диагностика обнаруживаемых погрешностей позволяет значительно сокращать сроки работ. В системе HYDRA имеются следующие диагностические возможности: 1) распечатка динамической памяти программы или ее отдельных участков в соответствии с задаваемыми условиями; 2) подсчет различного рода ситуаций, встречающихся при работе программы и характеризуемых определенными условиями; 3) анализ сложных ситуаций, встречающихся при работе программы и передачи управления модулям более высокого уровня в тех случаях, когда модуль, обнаруживший ту или иную ситуацию, не может «самостоятельно» найти правильное решение.

Распечатка динамически распределяемой памяти программы. Распечатка содержимого динамически распределяемой памяти программы или ее отдельных участков производится в системе с помощью системной подпрограммы DQSNAP, при обращении к кото-

рой в качестве параметров задаются идентификатор выдачи и цепочка BCD-символов, определяющих вариант печати, например:

CALL DQSNAP (HEADER, 12HLWMEFDCVØNS).

Здесь HEADER — идентификатор выдачи в BCD-форме, который занимает одно слово и используется в качестве заголовка последующей печати. Второй параметр — цепочка BCD-символов, которые определяют конкретные условия, интерпретируемые подпрограммой. Эти буквы-условия позволяют отбирать различные варианты распечатки и формат выдачи, например: выдачу системных связей и связей рабочей памяти (L), выдачу системных связей, связей и данных рабочей памяти (W), печать таблицы входов для каждого банка (M), печать таблицы входов всех связей каждого банка (E) и т. д.

Ограничителем цепочки условий является не заданный подпрограмме символ, в качестве которого обычно используется точка. Например, следующее обращение к подпрограмме

CALL DQSNAP (TEXT, 'LME')

приведет к выдаче на печать связей рабочей области памяти и картины памяти со всеми связями.

Печать таблицы памяти имеет своей целью дать всестороннюю информацию о всех находящихся в ней банках данных. На каждый банк имеется специальная строка, называемая «вход в таблицу», которая содержит следующие данные:

1) системный байт статусного слова;

2) байт пользователя в статусном слове;

3, 4) адрес статусного слова в памяти, восьмеричный и десятичный;

5) идентификатор банка в BCD-форме;

6—8) число связей, число структурных связей и число данных;

9) флаг: ** — банк отмечен как неиспользуемый;

● — все отличные от нуля связи печатаются;

+ — последняя напечатанная связь не является последней ненулевой связью банка;

10) связи 1, 2 . . . N печатаются в порядке их расположения в банке, где N — наименьшее значение из двух чисел: N1 — последняя ненулевая связь данного банка; N2 — число связей, которые можно напечатать на оставшейся длине строки (это зависит от ширины печатающего устройства и обычно равно восьми).

Таким образом, таблица памяти — это в то же самое время таблица наиболее интересных связей банка. Распечатка отдельного банка — это выдача на печать его содержимого в восьмеричном или переменном формате в зависимости от заданных условий. Каждое число печатается вместе со своим адресом. Связи, указы-

вающие истинные адреса блоков, печатаются вместе с их названиями в формате 5XA6, I5.

Распечатка рабочей области памяти — это выдача на печать системных связей и собственно рабочей памяти, или только связей, или связей и обычных данных в соответствии с заданными подпрограмме условиями.

Подсчет различного рода ситуаций, встречающихся в ходе обсчета данных. При анализе работы программы весьма полезно знать, насколько в процессе работы встречаются различного рода ситуации, характеризуемые определенными условиями. Для решения этой задачи в обычные программы включаются счетчики различного рода ситуаций и специальные подпрограммы занимаются их анализом.

Система HYDRA для решения такой задачи представляет в распоряжение пользователя набор специальных подпрограмм, объединенных в R-пакет (Recording package). Эти подпрограммы подсчитывают число появлений тех или иных ситуаций, встречающихся в процессе анализа данных и идентифицируемых соответствующими номерами. При этом значения счетчиков могут возрастать не только на единицу, но и на любое заданное целое число. Когда число обнаруженных ситуаций превышает заданное, то система может сгенерировать новое условие.

Номера условий ID, соответствующие определенным ситуациям, могут задаваться индивидуально или группами с помощью специальных блоков информации. Помимо номера условия системе можно сообщить дополнительную информацию в виде сообщений пользователю, которые накапливаются до тех пор, пока в динамической памяти имеется место для их записи.

При обнаружении ситуации, требующей обращения к подпрограммам R-пакета, вызывается соответствующая системная подпрограмма, которой в качестве параметра задается номер условия (от 1 до 999). Эта подпрограмма находит в динамической памяти соответствующий счетчик и увеличивает его на заданную величину.

Если при обращении к подпрограмме R-пакета переполняется отведенный для их записи участок памяти, система генерирует новое условие, номер которого определяется конкретной ситуацией. Например, банк сообщений полностью заполнен, переполнен участок памяти, отведенный для записи условий, которые идентифицируются программой, и т. д.

Передача управления модулям более высокого уровня. В некоторых случаях при обращении к подпрограммам R-пакета требуется не только зафиксировать появление той или иной ситуации, но и передать управление из данного модуля другому более высокого уровня. Необходимость передачи управления возникает в тех случаях, когда подпрограмма, обратившаяся к системе,

не может самостоятельно «решить вопрос» о дальнейшем варианте работы. В этих случаях системе указывается место, куда нужно передать управление. Она запоминает его и организует переход таким образом, что при последующих возникновениях такого рода ситуаций они будут фиксироваться вместе с соответствующим типом перехода.

Для передачи управления и регистрации обнаруженной ситуации вызывается системная подпрограмма RQTRAP, которой, наряду с другими данными, указывается место, куда необходимо передать управление. Эта подпрограмма запоминает ситуацию и устанавливает соответствующий ей тип перехода. Тип перехода определяется подпрограммой RQTRAP в соответствии с условиями, заданными в специальных блоках информации.

При любом обращении к подпрограммам R-пакета можно установить флаг, наличие которого требует передачи управления. В этих случаях после завершения работы системных подпрограмм управление будет передаваться модулям более высокого уровня.

Организация передачи управления представляет пользователям две следующие возможности:

1) установив типы переходов для встретившихся конкретных ситуаций, система позволяет подсчитывать их число в процессе обсчета данных;

2) обо всех неожиданных или интересных ситуациях, встречающихся в работе и не определенных заранее, можно сообщать подпрограммам R-пакета. Это позволяет установить, сколь часто те или иные ситуации имеют место, и использовать эту информацию для отладки программ.

Основные подпрограммы R-пакета. Засылка параметров подпрограмм R-пакета и создание соответствующих структур данных для обмена информацией между ними производятся подпрограммой RQINIT. Эта подпрограмма вызывается после вызова подпрограмм инициирующих работу M-, T- и J-пакетов (T-пакет предназначен для обработки блоков информации и его назначение будет подробно рассмотрено в следующем разделе).

Подпрограмма RQTELL (*ID*, *IFLAG*) используется для сообщения R-пакету о возникновении ситуации, идентифицируемой номером условия ID. Если значение второго параметра IFLAG = = 0, то управление будет передано вызвавшей его подпрограмме. Если же IFLAG = 1, то управление передается процессору более высокого уровня.

Подпрограмма RQTELM (*ID*, *IFLAG*, *LINK*, *IARRAY*) позволяет передать системе не только номер встретившегося условия, но и дополнительное сообщение пользователю об адресе связи (LINK) и данных (массив IARRAY), которые затем можно отредактировать.

Передача управления и создание типов переходов производятся с помощью следующей последовательности операторов:

```
CALL RQTRAP (ICLASS)
IF (IQUEST (1).NE. 1) GØTØ **.
```

Здесь ICLASS — номер «типа перехода»; ** — метка оператора, которому необходимо передать управление; IQUEST — массив, находящийся среди данных прямого доступа. При первом обращении к RQTRAP его первая ячейка IQUEST(1) — 0. При последующих обращениях к ней находится номер условия ID, обнаружение которого потребовало передачи управления.

Редактирование сообщений пользователю, которые передаются с помощью RQTELM, производится подпрограммой RQEDIT.

Подпрограмма RQEND предназначена для завершения работы подпрограмм R-пакета стандартным образом. Она проверяет структуру данных пакета и организует редактирование соответствующей информации.

Блоки информации для подпрограмм R-пакета. Детальная информация для управления процессом регистрации различных ситуаций задается в специальных блоках информации RQID, RQTR и RQBL. Эти блоки задаются пользователем и содержат следующие данные: максимальные значения счетчиков для подсчета числа различных ситуаций; уровни регистрации ситуаций, отличные от нуля (нулевой уровень соответствует концу счета); группы условий, объединенные в типы переходов; размеры банков структуры данных R-пакета.

Блок RQID определяет конкретные условия (ID) и содержит для каждого заданного условия следующие данные: номер условия ID; уровень регистрации; максимальное значение счетчика. Нулевое значение счетчика означает отсутствие ограничения.

Если этот блок не задан, то система устанавливает, что все уровни регистрации равны нулю, а максимальные значения счетчиков не ограничены.

Блоки RQTR определяют типы переходов. Каждому переходу соответствует свой RQTR-блок, в котором задается: номер типа перехода; уровень подсчета; максимальное значение счетчика; номера условий, требующих организации данного перехода. Если эти блоки не заданы, то система создает только один тип перехода с номером 1, соответствующий определенным системным условиям.

Блок RQBL определяет размеры банков структуры данных R-пакета, которые зависят от числа дополнительных записей для предварительно не определенных условий, числа записей, отведенных для подсчета условий, характеризуемых названиями моду-

лей, числа связей и данных, накапливаемых пакетом. Если этот банк не задан, то используются системные значения величин.

6. СИСТЕМНЫЕ ПОДПРОГРАММЫ ДЛЯ РАБОТЫ С БЛОКАМИ ИНФОРМАЦИИ

Модулям и ряду системных подпрограмм для работы требуются определенные данные, которые они находят в задаваемых пользователем блоках информации (Titles).

Специальные системные подпрограммы, объединенные в T-пакет, вводят эту информацию в ЭВМ, формируют из нее линейную структуру данных, а в процессе счета находят в ней нужные данные. Типичными примерами блоков информации являются для геометрических программ данные о параметрах установки и ее оптической системы, топография магнитного поля, критерии выбора хороших событий и т. п.

Блоки информации в BCD-форме вводятся с перфокарт или магнитной ленты подпрограммой TQINIT (LUN) (LUN — логический номер устройства). Затем из этих данных формируется специальная структура данных, называемая титульной структурой. Каждый банк структуры соответствует определенному блоку информации и снабжается соответствующим идентификатором, который позволяет системной подпрограмме QTITLE находить нужные блоки данных. Система позволяет также преобразовывать данные, задаваемые в блоках информации и создавать новые титульные банки. Для этих целей используется системная подпрограмма TQUSER, которая обеспечивает пользователю доступ к содержимому блоков информации и их преобразование. Поиск нужных программе банков титульной структуры производится с помощью системной подпрограммы QTITLE (L, ID, IFLAG).

Эта подпрограмма просматривает титульную структуру данных для нахождения в ней банка с идентификатором ID в первом слове и присваивает его адрес параметру L. Если она не находит банка с заданным идентификатором, то дальнейшие действия зависят от параметра IFLAG.

Когда IFLAG = 0, L = 0. Когда IFLAG = 1, то управление передается подпрограммам R-пакета. Идентификатор отсутствующего блока заносится в IQUEST (2). Подпрограмма QTITLE также перемещает найденный блок на первое место в структуре данных, что позволяет экономить машинное время при неоднократных обращениях к этому блоку.

Подпрограмму TQINIT (LUN) следует вызывать сразу же после вызова MQINIT, так как титульная структура данных должна быть создана и занесена в память до вызова тех системных пакетов, которые ее используют. Банки титульной структуры создаются системой в следующем формате:

	Идентификатор банка ID/NL/NS/ND
−1	Первая связь банка
LBK+0	Статусное слово
+1	Название банка
+2	Первое слово данных
	. . . . .
	. . . . .
	. . . . .
+ND	Последнее слово данных

Таким образом, если L — адрес банка, то Q (L + 1) — его название, Q (L + 2) — первое слово данных, Q (L + 3) — второе слово данных и т. д.

Исходные данные для титульных банков задаются следующим образом: каждому банку (или группе банков) структуры соответствует определенный блок информации. На первой карте блока задаются условия формирования банка, формат считывания блока информации и другие данные, которые располагаются в следующем порядке:

Столбец 1. * — признак первой карты блока информации.

Столбец 2. Пробел — для обычных блоков, * — для блоков с названиями частиц, N — (целое число от 0 до 9) — для блоков, преобразуемых программой в новые титульные банки.

Столбцы 3—4. ID — указатель на то, что первое слово данных следует использовать в качестве идентификатора для поиска блока.

Столбцы 5—6. Пробелы.

Столбцы 7—10. Четырехсимвольное название блока.

Столбцы 11—20. Число слов с данными.

Столбцы 21—30. Формат для считывания карт с данными.

7. СИСТЕМНЫЕ ПОДПРОГРАММЫ ДЛЯ ФОРМИРОВАНИЯ, НАКОПЛЕНИЯ И РАСПЕЧАТКИ ГИСТОГРАММ И ГРАФИКОВ

В системе HYDRA имеются специальные подпрограммы, объединенные соответственно в H- и P-пакеты, которые предназначены для формирования гистограмм и графиков, накопления их в динамической памяти и выдачи на печать.

Основные подпрограммы этих пакетов предназначены для выполнения следующих операций:

определения параметров для создаваемых гистограмм и графиков (подпрограммы QHISTO и QPLØTO);

создания в динамической памяти банков для накопления гистограмм и графиков (подпрограммы QHIST1 и QPLØT1);

засылки в банки гистограмм или графиков соответствующей информации (подпрограммы QHIST2, QHISTW и QPLØT2);

организации выдачи на печать всех накопленных или только заданных гистограмм и графиков (подпрограммы QHIST3 и QPLØT3).

Примечание. Здесь все подпрограммы, использующие в качестве второй буквы названия буквы H, относятся к H-пакету, а P — к P-пакету.

Гистограммы хранятся в специальных банках QH линейной структуры данных. Каждый банк может содержать несколько подобных гистограмм, объединенных в родственную группу. Все гистограммы одной группы имеют общее название и одни и те же параметры: число интервалов, ширину интервала, границы гистограммы и максимальное число событий в интервале. Внутри

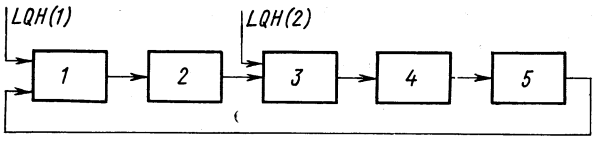


Рис. 7

банка гистограммы идентифицируются порядковыми номерами. При обращении к QHIST2 в качестве фактических параметров задаются следующие данные: название гистограммы, ее порядковый номер в банке и гистограммируемая величина.

Банки QH структуры данных создаются последовательно один за другим подпрограммой QHIST1 и объединяются в линейную циклическую структуру, сохраняющую порядок, в котором создавались банки. Специальная системная связь (LQH(1)) указывает адрес первого банка QH структуры, первая связь этого банка — адрес второго и т. д. Первая связь последнего банка указывает адрес первого и закрывает структуру (рис. 7).

После обращения к подпрограмме QHIST2 она ищет в структуре нужный банк и записывает найденный адрес в системную ячейку LQH (2). При новом обращении поиск нужного банка начинается с этой ячейки, и следует создавать гистограммы в том порядке, в котором программа будет заполнять соответствующие банки. Выполнение этого условия позволит экономить машинное время, затрачиваемое на создание и накапливание гистограмм.

В принципе возможны ситуации, когда программа пытается накапливать данные о гистограмме, которая не определена в банках. Хотя эту возможность стараются устранить, тем не менее она встречается. В таких случаях, чтобы избежать напрасных потерь времени на поиск нужного банка, система создает специальный пустой банк и включает его в структуру данных. Обращение к системе в этих случаях не имеет смысла, но наличие такого банка позволяет экономить время, затрачиваемое на поиск не определенных заранее гистограмм по всей структуре.

Организация выдачи на печать производится с помощью системной подпрограммы QHIST3. В качестве параметров, которые задаются при ее вызове, используются название гистограмм, их число в банке и цепочка BCD-символов, определяющих характер выдачи.

С помощью специальных символов можно организовать различные варианты выдачи, так, например, печать всех накопленных гистограмм, выборочных гистограмм, заголовки и суммарные данные о гистограммах (средние значения гистограммируемых величин, среднеквадратический разброс и т. п.), печатать на одной странице несколько гистограмм и т. д.

Реализация этих возможностей достигается с помощью специальных системных подпрограмм: QHFIND, QHLØAD, QHHEAD, QHSUM, QHPRNT.

Подпрограмма QHFIND (*NAME*, *JH*) находит банк с названием NAME, в котором накапливается гистограмма.

Подпрограмма QHLØAD создает массив в памяти, в который засылаются данные о создаваемой гистограмме.

Подпрограмма QHHEAD печатает заголовок гистограммы в BCD-форме, состоящий не более чем из 80 символов.

Подпрограмма QHSUM предназначена для печати суммарных результатов гистограммирования.

Подпрограмма QHPRNT организует выдачу гистограммы на печать.

Обмен служебной информацией между системными подпрограммами и программой проводится через специальные общие блоки системы. Подпрограммы P-пакета выполняют в основном те же функции, что и аналогичные подпрограммы H-пакета. CALL QPLØT0 определяет параметры создаваемых графиков. CALL QPLØT1 создает главный банк для группы подобных графиков, имеющих общее название. Разница с H-пакетом здесь заключается в том, что для каждого графика группы создается специальный подбанк, в который заносятся соответствующие данные (X- и Y-е координаты) и адрес которого определяется соответствующей связью главного банка.

CALL QPLØT2 создает требуемый подбанк и заполняет его данными.

CALL QPLØT3 организует печать графиков.

В заключение следует заметить что все, что говорилось ранее о порядке организации вызова подпрограмм H-пакета, также применимо и к подпрограммам P-пакета.

8. БИБЛИОТЕКА СИСТЕМЫ И ЕЕ ОРГАНИЗАЦИЯ

Библиотека системы HYDRA состоит из следующих частей:

1) группа подпрограмм общего назначения (GENERAL SECTION), предназначенных для работы с векторами и матрица-

ми, преобразования символов в целые числа и целых чисел в символы, операций с битами, байтами и т. п. В настоящее время в этой группе имеется около 110 подпрограмм, которые используются пакетами и модулями системы Для проверки работы этих подпрограмм в системе имеется специальная тестовая программа MICKY;

2) системные подпрограммы, объединенные в несколько пакетов и предназначенные для расширения основного алгоритмического языка. Эта часть является в узком смысле слова системой HYDRA;

3) наборы модулей, предназначенные для решения различного рода задач прикладного характера (восстановления пространственной картины событий, кинематического анализа результатов реконструкции и т. п.);

4) набор тестов для проверки системных пакетов и прикладных программ, составляемых из модулей системы, а также различного рода дополнительной информации, необходимой для постановки системы на ЭВМ.

Все элементы системы записываются на магнитную ленту в упакованном виде, и поэтому для ее чтения имеется специальная программа CETA. К магнитной ленте с библиотекой прилагается небольшая программа, которая позволяет получить с нее программу CETA.

Все элементы системы HYDRA, включая модули, хранятся на магнитных лентах или магнитных дисках в виде PAM-файлов [2, 8]. Сборка различных вариантов системы производится из элементов PAM-файлов с помощью специальной программы PATCHY [6].

Организация системы HYDRA в виде PAM-файлов является весьма удобным и эффективным аппаратом, позволяющим относительно легко вести работы по ее постановке на различных ЭВМ.

Элементы системы располагаются на библиотечной ленте в следующей последовательности: CETA, PATCHY, GENERAL SECTION, MICKY, HYDRA, USER PAM, GEOMetry PAM, UTIL PAM, KINEmatics PAM.

Все подпрограммы, которые зависят от конкретных характеристик вычислительной машины, выделены в специальные разделы PAM-файлов GENERAL SECTION и HYDRA.

В связи с тем что программа PATCHY зависит от конкретной ЭВМ и ее операционной системы, работу по постановке системы HYDRA необходимо начинать с создания версии программы PATCHY.

В PAM-файл пользователя (USER PAM) включены подпрограммы, в которых задаются разнообразные константы, зависящие от конкретных экспериментальных условий, режима выдачи на печать результатов анализа данных, набора констант для

ряда экспериментов и различных трековых камер. Наличие в системе этого PAM-файла облегчает пользователю сборку прикладных программ.

GEOM PAM содержит набор модулей, которые предназначены для восстановления пространственной картины событий в больших пузырьковых камерах и камерах классического типа.

В UTIL PAM собрана различного рода вспомогательная информация, необходимая для постановки системы и ее элементов на новой ЭВМ. К ней относятся блоки информации и описания банков данных, тестовые события, а также программа WRUP, предназначенная для получения документации о системе.

KINE PAM содержит набор модулей, предназначенных для кинематической идентификации событий на основе анализа законов сохранения энергии-импульса в вершинах событий.

9. ПОСТАНОВКА СИСТЕМЫ

Наиболее удобная форма работы с элементами системы HYDRA при ее постановке на ЭВМ — использование личной библиотеки [7] на магнитном диске или магнитной ленте. Личная библиотека различных элементов системы составляется из их PAM-файлов с помощью программы PATCHY. Для постановки системы на новой ЭВМ необходимо иметь библиотечную магнитную ленту с программой для ее чтения, программу PATCHY, а также результаты обсчета тестовых событий по различным подпрограммам системы.

Подготовка версии системы для новой вычислительной машины производится в следующей последовательности.

1. *Редактирование и сборка текстов подпрограмм общего назначения*, их трансляция и отладка с помощью тестовой программы MICKY. Так как почти все подпрограммы этой части написаны на ФОРТРАН, то их постановка не представляет особых затруднений. Подпрограммы, которые не могут быть реализованы на ФОРТРАН, выделены в отдельную секцию. К ним относятся подпрограммы для работы с битами, байтами, символами и т. п. Если для данной ЭВМ такой секции нет, то она создается. Кроме того, на автокоде рекомендуется писать также наиболее часто используемые подпрограммы для сокращения затрат машинного времени.

2. *Редактирование и сборка текстов системных подпрограмм*, трансляция и проверка на тестах системы. В процессе этой работы вносятся изменения в основной текст, обусловленные параметрами данной вычислительной машины (длина слова, число символов в слове, длина строки, распечатываемой на АЦПУ, форматы выдачи на печать и некоторые другие величины). Подпрограммы, предназначенные для передачи управления от одного модуля

к другому и в некоторых других сложных случаях составляются на автокоде. Затем на специальных тестах системы проводится проверка основных режимов работы системных подпрограмм.

3. *Редактирование и отладка модулей системы.* В рамках системы конкретные прикладные программы не являются ее элементами, а составляются из отдельных модулей. Так, геометрия системы состоит из набора модулей (GEOM PAM), предназначенных для решения основных задач процесса реконструкции пространственной картины событий: преобразования результатов обмера стереоснимков из одной координатной системы в другую и учета искажений; реконструкции вершин; идентификации проекций одних и тех же треков на стереоснимках события; реконструкции треков и определении их параметров.

Конкретные геометрические программы составляются из модулей, системных подпрограмм и подпрограмм общего назначения. Чтобы облегчить работу, большинство процедур, зависящих от характера экспериментальной установки, собраны в так называемые подпрограммы пользователя. Для отладки модулей системы с помощью подпрограмм пользователя составляется конкретный вариант прикладной программы, работа которого проверяется на тестовых и реальных событиях. Проверенные таким образом модули могут использоваться в других прикладных программах.

ЗАКЛЮЧЕНИЕ

Модульная система HYDRA в относительно короткий срок завоевала широкое признание и в настоящее время широко используется для создания программ обработки фильмовой информации и других экспериментальных данных, получаемых в исследованиях на ускорителях заряженных частиц.

Достаточно указать, что в совещаниях, проводимых в ЦЕРНе по этой системе, участвуют представители около двадцати крупнейших институтов стран Западной Европы. В настоящее время имеются версии этой системы для большого числа различных ЭВМ (СДС-7600, СДС серии 6000, IBM 360/75, IBM 370/195, UNIVAC-1108, РДР8, РДР10, БЭСМ-6 и др.).

Внедрение системы HYDRA в практику программирования не только расширяет возможности алгоритмического языка ФОРТРАН, но и значительно сокращает сроки работ по созданию математического обеспечения экспериментов и расширяет возможности сотрудничества в этой области между различными институтами.

В заключение следует отметить, что разработчики системы HYDRA много усилий потратили на то, чтобы ее адаптация на другие ЭВМ проходила как можно легче. В связи с этим имеются фортрановские тексты почти всех системных подпрограмм и тесты

для их проверки. Библиотека подпрограмм общего назначения также снабжена специальной тестовой программой. Система снабжена подробным руководством, в котором рассмотрена работа основных системных подпрограмм, общих блоков, используемых системой для обмена данными, и т. п. Все это позволяет производить постановку системы на достаточно мощных ЭВМ небольшими силами и в разумные сроки.

СПИСОК ЛИТЕРАТУРЫ

1. **Blair W. M. R.** Препринт ЦЕРНа, DD/DA/68/9, 1968.
2. **Program T. C.** Library, V. 1—3. ЦЕРН, 1968; **Буздавина Н. А. и др.** Препринт ОИЯИ, Р10-5785, 1971.
3. **Bock R. K. e.a.** Сообщение ОИЯИ Д10-6142, 1971, с. 547—564.
4. **HYDRA** Bulletin, ЦЕРН, 1973; **HYDRA** System Manual, ЦЕРН, 1973; **HYDRA** Application Manual, ЦЕРН, 1973.
5. **Говорун Н. Н. и др.** Сообщение ОИЯИ Д10-7707, 1974, с. 555.
6. **Дорж Л. и др.** Сообщение ОИЯИ 10-6882, 1973.
7. **Мазный Г. Л.** Сообщение ОИЯИ 11-5974, 1971.
8. **Буздавина Н. А. и др.** Сообщение ОИЯИ, 10-7192, 1973.