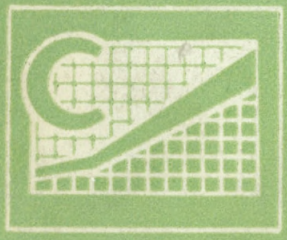
# Алгоритмы и организация решения экономических задач



# АЛГОРИТМЫ И ОРГАНИЗАЦИЯ РЕШЕНИЯ ЭКОНОМИЧЕСКИХ ЗАДАЧ



*Сборник статей*
*под редакцией*
*В. М. Савинкова*



Выпуск
13



МОСКВА
«Статистика»
1979

ББК 32.973
А45



А $\frac{30502-034}{008(01)-79}$ 72—79 1502000000

# ПРЕДИСЛОВИЕ

В статьях данного сборника рассматриваются новые тенденции в проектировании информационных систем и вопросы программного обеспечения автоматизированных банков данных (АБД). В связи с различием в толковании понятия «банк данных» и вследствие этого подменой на практике технологии АБД традиционной технологией работы на ЭВМ с единой информационной базой дадим определение автоматизированного банка данных.

АБД выступает как система программных, языковых, организационных и технических средств, предназначенных для централизованного накопления и коллективного использования данных, включая сами хранимые данные. Банк данных входит в обеспечивающую часть автоматизированной системы, реализуя информационную модель объектов управления. Архитектура АБД может рассматриваться в нескольких аспектах.

Банк данных как человеко-машинная система включает внешних и внутренних пользователей, администрацию банка данных, информационный процессор на базе ЭВМ (с требуемым количеством магнитных дисков), производящий необходимую обработку и пересылку требуемых данных, и, возможно, средства оперативного тиражирования (размножения) выходной информации. Внешний пользователь выступает в системе как потребитель информации, формулирующий задачу в терминах внешних объектов (профессиональные языки). Внутренний пользователь является поставщиком информации, непосредственно участвуя в технологическом процессе отбора, подготовки и ввода данных в АБД.

Банк данных как информационная система содержит фонд данных (базы данных, описания данных, словари, прикладные программы), управляемый системой управления банком данных (СУБнД). В составе СУБнД выделяют систему управления базами данных (СУБД) в качестве главного компонента. СУБД рассматриваются как основной результат в развитии программного обеспечения ЭВМ со времени появления операционных систем.

СУБД обеспечивают сокращение сроков проектирования информационных систем, их эффективную эксплуатацию и быструю адаптацию к постоянно изменяющимся условиям применения.

В 1974 г. Конгресс международной федерации по обработке информации ИФИП-74 определил основные тенденции развития банков данных следующим образом:

а) банки данных станут в основном распределенными, коллективного пользования;

б) языковые компоненты СУБД в большой мере будут основываться на математических моделях, например исчислении отношений;

в) в середине 90-х годов основным будет «случайный» пользователь, общающийся с системой на естественном языке, в отличие от параметрических * пользователей середины 80-х годов.

Ожидается, что через 10—15 лет индустрия обработки данных превратится в основную отрасль промышленности США. С началом новой «информационной» революции связывают банки данных, в которых стоимость создания, хранения и поддержки одного байта информации будет доведена до стоимости одного полиграфического знака при значительно больших возможностях последующего изменения и многократного автоматического использования.

Наибольшее распространение за рубежом, как известно, получили следующие СУБД, используемые в банках данных: MARK-IV, IDS, IMS, DBMS и др. В нашей стране создан ряд аналогичных и оригинальных СУБД: «Ока», «Банк», «Набоб» **, СХУД и др.

В информационных системах, создаваемых на основе СУБД, способы организации данных и методы доступа к ним перестали играть для пользователя решающую роль, поскольку оказались скрытыми внутри систем управления базами данных. Наилучшая же структура данных выбирается с помощью средств моделирования базы данных. Пользователь, как правило, имеет дело только с внешним интерфейсом — специальным языком его предметной области. В последнее время появились также системы (например, Robot), воспринимающие английский текст и связанные с СУБД. Такие системы позволяют формулировать проблему (задачу) на общедоступном языке.

Создание банков данных, рассчитанных на ежегодное поступление информации, объемом в миллионы символов, хранение этой информации и непосредственный доступ к ней сотен абонентов предъявляют следующие серьезные требования к техническим средствам:

наличие ЭВМ с эффективным быстродействием свыше миллиона операций в секунду, с оперативной памятью 1—2 Мбайта (а с учетом виртуальности до 16 Мбайт);

наличие памяти непосредственного доступа с общим объемом несколько миллиардов байт, для чего требуются большие диски с объемом хранимой информации до 100 Мбайт;

наличие разнообразных средств непосредственного общения або-

---

* Параметрический пользователь формулирует задачу обработки данных в терминах параметров систем программирования.

** См. статью Ф. Л. Фридлендера и В. М. Савинкова в предыдущем выпуске сборника.

нентов с ЭВМ и достаточно быстрых каналов связи как для ввода информации, так и для осуществления поиска информации;

наличие специальных средств автоматизации тиражирования различных документов для осуществления избирательного распределения информации.

Эксплуатация и обслуживание банков данных являются проблемой, превосходящей во много раз по сложности эксплуатацию и обслуживание ЭВМ. Решение этой проблемы при проектировании банков данных (проектировании технологии) невыполнимо при отсутствии хотя бы одного из вышеперечисленных компонентов.

В нашей стране наметились и интенсивно развиваются разработки банков данных в различных направлениях. Основное направление применения банков данных — автоматизированные системы управления технологическими процессами, предприятиями, объединениями и отраслями. Из межотраслевых систем, в рамках которых создаются банки данных, следует выделить экономические, в том числе и статистические, социально-политические банки данных, а также банки научно-технической информации. Имеют место работы над банками данных систем автоматизации проектирования и др.

Приведенная классификация основана на различиях таких характеристик банков данных, как вид поступающей информации, состав языковых средств пользователей, требования на время и полноту обработки и др.

Продолжая в настоящем выпуске публикации по указанной тематике редакционная коллегия предполагает в очередных выпусках ознакомить читателей с опытом применения СУБД при проектировании информационных систем.

# I МЕТОДОЛОГИЯ И ОРГАНИЗАЦИЯ РЕШЕНИЯ ЭКОНОМИЧЕСКИХ ЗАДАЧ

*В. А. МЯСНИКОВ*

## ЗАДАЧИ И ПЕРСПЕКТИВЫ РАЗВИТИЯ ПРОИЗВОДСТВА И ИСПОЛЬЗОВАНИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ В НАРОДНОМ ХОЗЯЙСТВЕ СТРАНЫ

### ОБЩЕГОСУДАРСТВЕННАЯ АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ

В «Основных направлениях развития народного хозяйства СССР на 1976—1980 годы» записано: «Обеспечить дальнейшее развитие и повышение эффективности автоматизированных систем управления и вычислительных центров, последовательно объединяя их в единую общегосударственную систему сбора и обработки информации для учета, планирования и управления. Создавать вычислительные центры коллективного пользования».

Директивами XXIV съезда КПСС была поставлена задача развернуть работы по созданию и внедрению автоматизированных систем управления на уровне отраслей, объединений и предприятий с постепенным переходом к общегосударственной автоматизированной системе (ОГАС) на базе государственной сети вычислительных центров и общегосударственной системы передачи данных.

В соответствии с Директивами XXIV съезда КПСС были начаты работы по проектированию и внедрению АСУ различных уровней, налажен серийный выпуск ЭВМ третьего поколения (на интегральных схемах).

Экономико-математические методы и ЭВМ нашли широкое применение для совершенствования планирования и оперативной деятельности в отраслях (ОАСУ), объединениях и крупных предприятиях (АСУП).

В текущей пятилетке для ОАСУ, АСУП, научных, проектных организаций и других целей будет выпущено несколько тысяч универсальных ЭВМ третьего поколения. Взят курс на создание мощных ЭВМ (с быстродействием 5, 10 и более 100 млн. оп./с) и управляющих ЭВМ, что является оптимальным и отвечает мировым тенденциям.

Вычислительная техника стала неотъемлемой частью автоматизации сложных технологических процессов в химии, нефтехимии, энергетике, черной металлургии и других отраслях. Для этих и других целей в X пятилетке Минприбор и Минэлектронпром выпустят несколько тысяч управляющих ЭВМ.

Анализ потребности в ЭВМ показывает, что удовлетворение пользователей управляющими и универсальными ЭВМ осуществляется не полностью. Вычислительные центры (а их сейчас насчитывается в стране около 3000) пока что недостаточно оснащены периферийным оборудованием (быстропечатающими устройствами, накопителями на магнитных дисках и лентах и т. д.), имеется недостаток в аппаратуре передачи данных, неблагополучно обстоит дело с эксплуатационными материалами (бумажными и магнитными носителями информации и др.).

Эти и другие недостатки (неподготовленность задач у пользователей, неготовность помещений и т. п.) сдерживают эффективное использование ВТ и, в частности, приводят к тому, что в среднем загрузка в стране ЭВМ составляет около 12 ч вместо 16—18 ч в сутки.

Несмотря на эти недостатки, эффективность капиталовложений в ВТ и АСУ выше, чем по другим направлениям капиталовложений. На каждый рубль капитальных вложений в объекты ВТ в IX пятилетке была заложена плановая экономия в размере 33 коп. (коэффициент эффективности 0,33), фактически же этот коэффициент составил 0,4. Экономия от внедрения ВТ в IX пятилетке составила около 1,8 млрд. руб. В текущей пятилетке за счет работы АСУ и ВЦ должна быть получена экономия в 3,8 млрд. руб., из них около 2 млрд. руб. — благодаря снижению себестоимости продукции, что учитывается в разделе «Прибыль и себестоимость» нархозплана.

Проблема улучшения планирования и управления народным хозяйством на данном этапе развития экономики и технического прогресса может и должна решаться путем проведения целого ряда организационных и технических мероприятий.

Организационные мероприятия охватывают совершенствование управления отраслями, ликвидацию многозвенности, создание объединений. Однако данные мероприятия не могут в полной мере обеспечить решение всех задач, поставленных ЦК КПСС в области совершенствования планирования и управления.

Количественный рост производства продукции и качественные изменения технологических процессов сопровождаются увеличением информации, необходимой для оптимального планирования и управления материальным производством. Объем циркулирующей информации в народном хозяйстве эквивалентен примерно 25 миллионам книг по 500 страниц каждая. В стране ежегодно зарождается около 60 млрд. письменных документов. Оценки показывают, что к 1990 г. объем информации, необходимой для планирования и управления, возрастет в 2—3 раза.

Дело складывается таким образом, что через 15—20 лет коренным образом изменится вся система сбора, обработки и хранения

информационных данных, необходимых для плановых расчетов и управленческих решений. АСУ на базе ЭВМ являются единственно возможными техническими средствами для успешного совершенствования процессов управления; при этом разумеется, что экономистами и математиками должны совершенствоваться соответствующие модели экономики, методы балансовых расчетов и т. д.

Автоматический сбор, передача и обработка информации, осуществляемые с помощью ЭВМ и периферийных устройств, приведут к резкому сокращению, а в некоторых областях к полному исчезновению принятой сейчас отчетности и переписки. Справочные, нормативные и расчетные материалы будут храниться в запоминающих устройствах ЭВМ. При этом информация по запросу может выдаваться как в виде письменного документа, так и визуально на телевизионном экране. Таким образом, изменится вся система подготовки и формирования деловой документации. К этому следует добавить, что и производственная документация (чертежи, технологические карты, спецификации, проектные расчеты и т. д.) также может быть изготовлена в АСУ с помощью ЭВМ (что уже и сейчас делается в некоторых отраслях промышленности).

Все вышеизложенное лишний раз убеждает в правильности решений XXIV съезда КПСС о совершенствовании управления на основе широкого использования экономико-математических методов, АСУ и ВТ.

Решения XXV съезда КПСС подтверждают это направление и вместе с тем обязывают повышать эффективность АСУ и ВТ, осуществлять меры по последовательному объединению АСУ в единую общегосударственную систему, вводить новую форму использования ЭВМ — создавать вычислительные центры коллективного пользования (ВЦКП).

Теперь необходимо дать определение, что же такое ОГАС? Кому она принадлежит? Кто ею пользуется и зачем эта система нужна.

**Итак, ОГАС является взаимоувязанной на принципах организационного, методологического и технического единства комплексной системой функционирования автоматизированных систем управления центральных общегосударственных органов (Госплана СССР, Госснаба СССР, ЦСУ СССР и др.), АСУ отраслей, республик, объединений, предприятий и организаций.**

Из этого определения следует, что ОГАС не «надстройка», не нечто возвышающееся над существующими органами управления, не новый экономический орган управления, а механизм, обеспечивающий совместную работу многих АСУ.

ОГАС не будет подменять и заменять существующие в стране органы управления, а должна являться средством, повышающим эффективность работы этих органов при совместном функционировании их автоматизированных систем.

ОГАС будет позволять всем государственным органам автоматизированно получать и выдавать через ВЦ своих АСУ необходимую информацию для решения межотраслевых и междуведомственных задач.

Функционирующие совместно с ОГАС автоматизированные системы управления дадут возможность решать следующие задачи:

разработку вариантов проектов перспективных и текущих планов развития народного хозяйства СССР, отраслей, экономики союзных республик и экономических районов страны, материально-технического снабжения, капитального строительства, транспортных перевозок, финансово-кредитных планов, трудовых ресурсов, ценообразования и др. Например, сейчас уже проводятся работы по началу совместного функционирования АСПР Госплана СССР с ОАСУ Минприбора и ОАСУ Минвнешторга — этим самым закладываются основы ОГАС;

разработку проектов программ развития экономики, науки и техники;

анализ тенденций развития социалистической экономики отдельных отраслей народного хозяйства, союзных республик и экономических районов страны;

учет и статистическую отчетность о результатах работы всех звеньев народного хозяйства в необходимом темпе;

упреждающий аналитический контроль за выполнением постановлений директивных органов.

Преимуществом такой системы, как ОГАС на базе сети вычислительных центров, являются информационные связи. Многие решения по управлению экономикой связаны со сбором и передачей данных с многих мест, расположенных в различных точках страны. Однако отдельные локальные автоматизированные системы не могут выполнить подобные требования: каждодневное взаимодействие между производственными планами заводов, планами поставок и перевозок сырья и потребностями потребителей. Эти взаимодействия сейчас обеспечиваются системами прохождения вторичных посылок, которые дороги, неудобны, подвержены ошибкам и к тому же медленны.

Для функционирования ОГАС, так же как и ее технической базы — ГСВЦ и ОГСПД, потребуется создать службу диспетчеризации с мощным программным обеспечением для управления вычислительной сетью. Эти программы-диспетчеры будут по запросам осуществлять подключение вычислительных центров автоматизированных систем друг к другу для совместной работы.

В Москве должен быть создан Главный вычислительный диспетчерский центр для коммутации ВЦ автоматизированных систем общегосударственных органов, отраслей и ведомств.

Естественно, что это один из возможных вариантов технического решения. Могут быть и другие: например, сама сеть передачи данных (ОГСПД) будет осуществлять коммутацию пакетов и каналов при совместной работе ВЦ и т. д.

Для обеспечения организационного, методологического и технического единств АСУ, которые должны взаимодействовать в ОГАС, уже в IX пятилетке проделана значительная работа.

Во-первых, организационное и методологическое единство достигается едиными принципами создания АСУ, структурой форми-

рования информационных данных, единством представления технико-экономической информации, унифицированной системой документации.

С этой целью Госкомитет по науке и технике совместно с Минприбором и другими министерствами и ведомствами разработал руководящие материалы и типовые проектные решения как в целом по ОАСУ и АСУП, так и по отдельным их подсистемам.

Во-вторых, Госпланом СССР, Госстандартом СССР, министерствами и ведомствами разработаны 13 унифицированных систем документации и Единая система классификации и кодирования технико-экономической информации, включая 20 общесоюзных классификаторов. Все это должно обеспечить единство информационного «языка» между различными АСУ.

В-третьих, техническое единство достигается использованием ЭВМ Единой системы, которые программно и технически совместимы друг с другом.

Как показали работа над ОГАС и проектирование ГСВЦ, капитальные затраты на создание АСУ и ВЦ в народном хозяйстве СССР с учетом их объединения в ОГАС на основе ГСВЦ и ОГСПД могут быть снижены в два и более раза. Кроме того, будет получена существенная (в 2—2,5 раза) экономия на эксплуатационных расходах вычислительных центров.

Заслуживает внимания предложение поручить Госкомитету по науке и технике на период разработки и опытной эксплуатации выступить заказчиком, а следовательно, и «владельцем» ОГАС и ГСВЦ. Уточним, чем «владеть» конкретно: это информационные банки данных и межотраслевые ВЦКП, являющиеся технической базой ГСВЦ. В будущем все указанные функции могли бы быть переданы, например, вновь созданному Комитету информации, статистики и совершенствования управления. Разумеется, это очень сложные вопросы, требующие всестороннего рассмотрения, но решать их надо и чем быстрее, тем будет лучше для наведения четкого порядка в развитии ВЦКП, ГСВЦ, ОГСПД и в целом ОГАС.

Заказчиком и «владельцем» ОГСПД должно выступать Министерство связи СССР, так как у нас в стране осуществлена и проводится правильная политика централизованного предоставления услуг в области связи (телеграф, телефон и др.).

Остановимся на некоторых проблемах создания ГСВЦ и ОГСПД, которые являются технической базой ОГАС.

В целях упорядочения работ по созданию АСУ и ВЦ Госкомитету по науке и технике было поручено начать проектирование Государственной сети вычислительных центров как технической базы ОГАС.

Госкомитет по науке и технике и Всесоюзный научно-исследовательский институт проблем организации и управления с привлечением более 60 организаций других министерств и ведомств в течение 1973—1975 гг. разработали технико-экономическое обоснование, эскизный и рабочий проекты ГСВЦ. В 1976 г. Государствен-

ная экспертная комиссия рассмотрела и приняла указанные документы.

ГСВЦ будет представлять совокупность вычислительных центров страны, объединенных с помощью ОГСПД в вычислительную систему. Расчеты показывают, что для ее создания потребуется около 200 крупных ВЦКП, расположенных в различных точках страны. Вычислительные центры коллективного пользования основаны на использовании ЭВМ, обладающих свойствами одновременного автоматического обслуживания многих десятков и сотен пользователей и находящихся на значительном расстоянии от ВЦ (за сотни и тысячи километров). К этим ВЦКП будут постепенно подключаться вычислительные центры, расположенные в данном экономическом районе, в свою очередь должна быть организована связь ВЦКП между собой.

Следует особо подчеркнуть, что ВЦКП должны предоставлять пользователям не только машинное время, но и программный сервис.

Создание сети ВЦ, объединенных системами связи, является новой прогрессивной формой использования вычислительной техники, так как в этом случае вычислительными мощностями могут пользоваться многие пользователи (заводы, НИИ, КБ, вузы и др.). Сеть вычислительных центров по сравнению с индивидуальным использованием ЭВМ имеет значительные преимущества в экономическом и техническом отношениях.

Прежде всего в сети ВЦ электронные машины загружены максимально, а стоимость хранения и обработки информации, как отмечалось, снижается в 2—2,5 раза. Капитальные вложения уменьшаются более чем в 2 раза: пользователь не строит здание под ВЦ, не приобретает ЭВМ, не содержит обслуживающий персонал и т. д., а устанавливает терминальное оборудование — устройство ввода-вывода информации, соединенное линией связи с сетью ВЦ. В качестве этого устройства может применяться малая ЭВМ, к которой в свою очередь могут присоединяться более простые терминальные устройства. Создание ГСВЦ имеет еще одно важное преимущество — увеличивается надежность обслуживания при остановке одной или нескольких ЭВМ, так как в этом случае задачи передаются для решения другим ЭВМ сети. Как видно, в этом есть внешняя аналогия с энергетическими сетями. Сети ВЦ должны обеспечивать пользователей возможностью доступа к банкам данных и программ.

Можно отметить, что во многих развитых зарубежных странах вычислительные сети действуют давно. Например, в США имеется около 2000 ВЦКП и 250 вычислительных сетей.

В разработанных проектах ГСВЦ предусматривается, что существующие вычислительные центры АСУ отраслей, ведомств, предприятий и т. д., подключаясь и взаимодействуя между собой в ГСВЦ, остаются в полной собственности и административном подчинении соответствующих организаций.

В техническом проекте ГСВЦ на основе расчетов обосновыва-

ются структура ВЦКП (от 0,5 млн. до 50 млн. оп./с), система связи, требования к программному обеспечению и т. д. В нем также определены этапы создания ГСВЦ. На первом этапе должны быть созданы экспериментальные ВЦКП в городах Ленинград, Минск, Таллин, Томск, Тула, Рига, Тюмень и др.

Второй этап должен характеризоваться переходом от создания маломощных ВЦКП к более производительным с одновременным обслуживанием до 1000 и более пользователей. К концу этого этапа должно быть создано 50—60 таких ВЦКП, объединенных линиями связи. Суммарная производительность всех ВЦКП должна достигнуть 400 млн. оп./с. Сеть ВЦКП в качестве абонентов будет включать около 10 тыс. ВЦ индивидуального пользования министерств, ведомств и предприятий.

На третьем этапе в основном должно быть завершено создание ГСВЦ как технической базы ОГАС — около 200 ВЦКП с объединением их линиями передачи данных, а также организовано регулярное функционирование ВЦКП со всеми абонентами ГСВЦ. Можно выделить четыре уровня ОГАС и соответственно ГСВЦ.

**Первый уровень** — основу составляют автоматизированные системы общегосударственных организаций (Госплан СССР, Госснаб и Госстрой СССР, ГКНТ и др.), Академии наук СССР. Кроме того, на данном уровне должны располагаться отраслевые автоматизированные системы управления промышленностью, транспортом и связью, строительством, сельским хозяйством, а также системы непроизводственной сферы.

Технической базой, объединяющей все системы первого уровня ОГАС, может быть Главный вычислительный центр ОГАС, выполняющий также функции координирующего диспетчерского центра. Координирующие функции можно реализовать и в других вариантах, о чем говорилось выше.

**Второй уровень** — автоматизированные системы управления союзных республик и республиканских организаций (РАСУ). РАСУ — совокупность совместно взаимодействующих АСУ центральных органов республики, АСУ отраслями народного хозяйства республики и территориальными комплексами. РАСУ — это в миниатюре ОГАС. Всего РАСУ будет 15 — по числу союзных республик. Условно принято РАСУ разделять:

на АСУ республик, не имеющие областного деления (Молдавия и Прибалтика — Литва, Латвия, Эстония);

на АСУ союзных республик, имеющие в своем составе области, но компактные по занимаемой территории.

**Третий уровень** — территориальные вычислительные центры (ТВЦ). Они призваны выполнять следующие функции:

сбор и обработку информации через каналы связи по экономическим районам обслуживаемой территории;

обеспечение средствами вычислительной техники тех пользователей, которым экономически нецелесообразно иметь свои ВЦ;

осуществление связи начального уровня (заводы, предприятия,

вузы и т. д.) с первым уровнем — ГСВЦ, ОГАС, ГВЦ общегосударственных органов, отраслей и республики;

уплотнение, кодирование и шифрование информационных данных для передачи их в верхний уровень АСУ, входящий в ОГАС (при том, что ТВЦ могут быть совмещены с центрами коммутации сообщений);

быть резервом ВЦ высшего уровня.

Особенностью ТВЦ является необходимость сбора и хранения большого количества информации с последующей ее обработкой и подготовкой к передаче в ОГАС и через нее межведомственным и отраслевым ВЦ.

В СССР имеются 127 краев и областей и 20 автономных республик. Общее число вычислительных центров всех четырех уровней ОГАС будет превышать 25 тыс. единиц. Один вычислительный центр должен будет обслуживать в среднем 24 предприятия и организации.

**Четвертый уровень** — автоматизированные системы управления объединений, производственных предприятий и организаций.

В рассмотренной схеме существенное значение имеют единая автоматизированная система связи (ЕАСС) и общегосударственная система передачи данных. Единая система связи призвана обеспечить прямое и обратное взаимодействие между собой управляющих и управляемых организаций всех уровней, связанных каналами по вертикали и горизонтали.

## ОБЩЕГОСУДАРСТВЕННАЯ СИСТЕМА ПЕРЕДАЧИ ДАННЫХ

ОГСПД выполняет следующие функции:

сбор и уплотнение информации в центрах коммутации сообщений (ЦКС);

передачу информации по специальным каналам связи большой пропускной способности и скорости передачи в ВЦ верхних уровней ОГАС.

ОГСПД должна состоять из линий связи с пропускной способностью от 200 до 48 000 бод (бит/с), центров коммутации каналов и сообщений, технических средств передачи данных и т. д.

ЦКС будет содержать специальные устройства по уплотнению, кодированию, а при необходимости шифрованию информации. Они в первую очередь будут создаваться в крупных городах, а также в ВЦКП.

При создании ОГСПД необходимо предусматривать объединение разрозненно действующих сетей связи различных хозяйственных органов с целью повышения их загрузки. Достоверность передачи информации ОГСПД должна составлять $10^{-7}$.

Для создания ОГСПД частично можно использовать уже существующую систему связи. Однако многие задачи необходимо решать заново. Должна быть построена новая система центров коммутаций пакетов, каналов и сообщений и увеличена пропускная способ-

ность линий связи как за счет расширения старых, так и строительства новых линий.

Остановимся на перспективах и задачах развития дистанционной обработки данных. Реализация методов дистанционной обработки данных с помощью ЭВМ невозможна без самого широкого развития сетей связи и передачи данных. В принципе передача данных может осуществляться по обычным телефонным и телеграфным сетям общего пользования. Однако использование их для этой цели быстро становится неэффективным, так как они спроектированы для передачи телефонных и телеграфных сигналов и сообщений, а не данных для ЭВМ.

Рост потребностей в передаче и дистанционной обработке данных выявил значительное отставание техники связи от уровня развития вычислительной техники. Одной из важнейших возможностей уменьшения такого отставания является создание сетей, специализированных на передаче данных. К числу наиболее современных, обладающих большими перспективами и позволяющих использовать ЭВМ и средства телеобработки данных, могут быть отнесены сети с коммутацией сообщений (пакетов сообщений). Характерной особенностью таких сетей является наличие узлов коммутации, базирующихся на ЭВМ, процессорах связи. При этом узлы коммутации могут быть разного уровня. Например, группы абонентов подключаются к узлам коммутации нижнего уровня, которые выходят на узлы коммутации верхнего уровня. Последние соединяются между собой и с вычислительными центрами.

Применение ЭВМ в центрах коммутации позволяет обеспечивать высокую надежность, достоверность передачи информации без увеличения суммарного времени работы абонента для получения ответа. Наряду с этим может быть реализовано перераспределение потоков информации между отдельными ВЦ, по различным направлениям (при перегрузках или выходе из строя) и т. д.

Как указывалось, в СССР создается общегосударственная сеть передачи данных, в задачи которой входит передача данных для широкого круга пользователей вычислительной техники. Эта сеть включает центры коммутации каналов и сообщений, каналы связи, аппаратуру передачи данных.

Одним из важнейших факторов, определяющих создание и внедрение автоматизированных систем, является коллективное использование вычислительных ресурсов, ЭВМ, средств связи, передачи и телеобработки данных, техническая и экономическая целесообразность которого особенно заметно проявилась после интеграции перечисленных средств в единый организационно-технический комплекс.

Первым итогом такой интеграции должно быть создание ВЦКП. Такой центр может быть оснащен одной или несколькими ЭВМ равной (разной) производительности, к которым получают доступ десятки (сотни) удаленных и локальных абонентов. Концентрация вычислительных мощностей, наличие простых и эффективных средств доступа к ЭВМ (с помощью средств телеобработки данных)

обеспечивают большому числу удаленных абонентов возможность применения ЭВМ для решения конкретных задач обработки и хранения данных.

Следующим шагом является создание вычислительных систем коллективного пользования (ВСКП), сетей взаимосвязанных ЭВМ, включающих сотни и тысячи локальных и особенно удаленных абонентов. Такие сети обладают рядом достоинств: обеспечивают решение широкого круга задач, обработку значительных массивов данных, создание и хранение банков данных, программ и др.; обладают различной топологией и структурой (централизованная и децентрализованная, иерархическая и др.) и работают в различных режимах (диалоговом, пакетном, смешанном и т. д.).

Создание сетей ЭВМ требует системного подхода к разработке базовых элементов и в первую очередь средств передачи и телеобработки данных, ибо с их помощью объединяются средства связи и вычислительной техники, существенно отличающиеся по принципам и техническим параметрам.

Системный подход требует взаимосвязанного развития всех средств. Разработка средств передачи и телеобработки данных должна осуществляться в первую очередь с учетом их применения в сетях ЭВМ и сетях передачи данных. Невозможно обеспечить высокую эффективность ВСКП, если не достигнуто взаимодействие отдельных ее элементов или недостаточна пропускная способность сети передачи данных.

Системный подход заключается также в учете требований потенциальных пользователей как к ЭВМ, так и к средствам передачи и телеобработки данных (требования ко времени передачи сообщения, пакета сообщений от источника к пользователю, к достоверности, секретности передачи; исключение несанкционированного доступа, вероятности возникновения «пиков» нагрузки; ограничение стоимости установки и эксплуатации оборудования, каналов связи и т. д.).

Системный подход предусматривает разработку стандартизованных алгоритмов (протоколов) связи между ЭВМ, между ЭВМ и абонентскими пунктами (АП). Эти алгоритмы должны обеспечить адаптируемость новых типов оборудования (в первую очередь абонентских пунктов) к действующим вычислительным системам (ВС). С их помощью решаются задачи объединения ЭВМ различных серий (типов), производительности и с различными системами команд.

Важная роль в сетях ЭВМ отводится процессорам связи, которые реализуют подключение ЭВМ к сетям и их перераспределение по отдельным направлениям, подключение и управление работой АП. Разработка процессоров должна осуществляться по одним требованиям с базовыми ЭВМ, с общим программным обеспечением.

Проектирование сетей ЭВМ, развитие дистанционной обработки данных связаны с необходимостью решения некоторых основных задач теоретического и прикладного характера:

выбора критериев оценки эффективности функционирования сети (с учетом пожеланий пользователей и имеющихся ресурсов);

определения структуры (топологии) сети как функции ряда параметров (стоимость, надежность, назначение, интенсивность потоков информации, задачи обработки и др.); синтеза оптимальных конфигураций сети по заданным характеристикам и параметрам (разработки методов автоматизации выбора оптимальных конфигураций сети);

определения допустимого времени реакции (времени обработки и/или передачи данных); анализа времени прохождения сообщения (пакета сообщений) по сети, его нахождения в том или ином узле коммутации;

выбора маршрутов передачи сообщений при различных условиях работы сети (при нормальных условиях, возникновении «пиков» нагрузки, выходе из строя отдельных элементов сети);

анализа сети как системы массового обслуживания.

В процессе решения подобных задач общего характера должны быть решены и некоторые частные вопросы:

расширение услуг по обработке данных и предоставление пользователям дешевых, эффективных и простых в обращении средств связи с сетью (доступа к сети);

определение структуры и формата передаваемых сообщений с учетом возможных связей между ЭВМ различных типов и АП;

определение требуемых объемов буферной памяти в процессорах связи, программируемых и других типах АП и т. д.;

формирование и обработка очередей сообщений.

Актуальность решения этих задач подчеркивается значительным количественным ростом средств для дистанционной обработки информации. В печати приводится множество данных, характеризующих такой рост. В частности, система связи США, модернизированная в соответствии с требованиями и задачами данных, к 1980 г. должна обеспечить взаимодействие 200 тыс. ЭВМ и 10 млн. терминалов. Ожидаемое число терминалов в странах Западной Европы в 1980 г. 437,4 тыс. (в 1972 г. — 79,6 тыс., 1976 — 235,6 тыс.), а в мире — около 4 млн. При этом уже в 1978 г. число ЭВМ, работающих с удаленными абонентами, достигает 75% (1970 г. — 16%, 1975 г. — 60%).

В целом с учетом состояния разработки и производства отечественных средств передачи и телеобработки данных, тенденций развития вычислительной техники и автоматизированных систем можно указать первоочередные задачи создания перспективных систем передачи и обработки информации, включающие разработку:

процессоров связи, ориентированных на работу в ВСКП с учетом организации последних, в первую очередь на базе ЕС ЭВМ. При этом процессоры связи должны обеспечивать межмашинный обмен по высокоскоростным каналам, возможность применения в центрах коммутации сообщений, подключение сотен абонентов к системе;

программируемых абонентских пунктов, совместимых со средствами ЕС ЭВМ и СМ ЭВМ;

абонентских пунктов на базе устройств речевого ввода и вывода информации, устройств микрофильмирования, цветных и объемных индикаторов и др.;

комплекса аппаратуры передачи данных со скоростью работы более 500 тыс. бит/с.

Существенное значение имеют также повышение объема серийного производства и дальнейшее развитие сетей передачи данных на базе телефонных и телеграфных каналов и аппаратуры связи.

## ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОГРАММИРОВАНИЯ

Рассмотрим далее такие важные вопросы, как методы и средства повышения производительности труда программистов и состояние работ по созданию новых инструментальных средств технологии программирования.

Есть два общепризнанных пути повышения производительности труда программистов: создание хорошего языка программирования — языка обращения к ЭВМ и организации и упорядочение труда самого программиста в некоторой технологии программирования, регламентирующей высокую профессиональную культуру и грамотность работы исполнителей. Начиная с 1970 г. в СССР и за рубежом распространение получил второй путь как более эффективный на современном этапе развития вычислительной техники.

По данным института Шервуда (ФРГ) в 1971—1972 гг. суммарные затраты на программирование сравнялись с затратами на изготовление аппаратуры ЭВМ. В настоящее время затраты на программирование в 2—3 раза больше, чем на аппаратуру, и в абсолютном значении составляют несколько десятков миллиардов долларов. По зарубежным прогнозам на 1985 г. соотношение между стоимостью программного обеспечения и аппаратуры ЭВМ будет такое же, как между стоимостью товара и упаковки.

По времени разработки имеют место следующие данные. Время конструкторской разработки малой управляющей ЭВМ составляет в среднем 4—5 месяцев, разработка опытного образца — 6 месяцев, а программного обеспечения — от 3 до 5 лет (в США — от 2 до 4 лет).

Годовой объем производства программной продукции составляет в среднем 1000 млн. команд в США. Годовой прирост объема программной продукции в СССР и США равен в среднем около 15% при потребности 23%.

Производительность труда в СССР составляет в среднем 4—6 команд в день на человека, а в США — 8—10 команд; для систем реального времени указанные цифры составляют соответственно 1—1,5 и 1,4—2,7 команды в день на человека.

Таким образом, производство программной продукции является серьезной актуальной проблемой.

Одним из направлений совершенствования программирования выступает развитие алгоритмических языков.

«Погоня» за новыми языками, универсальным языком, языками-монстрами прекратилась. Четыре языка (Фортран, Кобол, ПЛ/1 и Паскаль) контролируют 85% применений языков неассемблерного уровня. Большое внимание уделяется модернизации этих языков: структурный Фортран, мета-Кобол, упрощенные диалекты ПЛ/1, РВ — Паскаль и т. д. Для модернизации строятся метасистемы, предпроцессоры, макрогенераторы и т. д., предназначаемые для универсальной проблемной ориентации существующих языков силами самих пользователей. К этому же направлению относится разработка пакетов программ с очень простыми (кнопочного типа или непроцедурными) языками обращения.

Особо следует выделить пакеты типа ПРИЗ, разработанного в 1976—1977 гг. Институтом кибернетики АН Эстонской ССР, которые можно рассматривать как универсальные инструментальные системы синтеза прикладных программ. Смысл этих пакетов состоит в том, что процесс построения (синтеза) программ делится на две части: одна из них (большая) задает модель задачи, а вторая — ее непроцедурное описание. Модель описывается высококвалифицированным профессиональным программистом один раз для целого класса применений. Если модель составлена тщательно, то вторая часть задачи — описание условия конкретной задачи, описание, что надо сделать (а не как и в какой последовательности, какими операторами), — может быть описана непрофессиональным пользователем на языке, близком к естественному. При этом подготовка непрофессионального пользователя может быть сведена к минимуму. По описанию условия задачи и ее модели система автоматически синтезирует нужную последовательность команд ЭВМ. Очевидно, что такие системы, должным образом изготовленные, весьма перспективны для прикладных применений.

Другое направление развития языков программирования характеризуется использованием абстрактных типов данных (языки CLU, АЛЬФАРД). Это направление рекламируется США как последнее достижение в программировании во второй половине 70-х годов. Суть его заключается в следующем. В языках Алгол, Фортран типы данных фиксированы (неизменные). В языках ПЛ/1, Паскаль пользователь может задавать свои типы и возможны сложные иерархические структуры. В перечисленных языках все операторы языка определены над всеми типами данных некоторым универсальным способом. В банках данных обработка данных отделена от процесса определения данных благодаря разделению языка манипулирования данными от языка описания данных. Последнее можно рассматривать как универсальный механизм формирования некоторых типов данных. Опыт эксплуатации именно банков данных выявил главный недостаток такого определения типов данных — универсальность определений делает системы сложными в освоении и малоэффективными в работе (по быстродействию и использованию ресурсов ЭВМ). На практике обычно структуры данных прос-

ты и над ними определен весьма ограниченный набор операций. Это свойство использовано в новом направлении. В нем пользователю предоставляется не только возможность определить свои специальные типы данных, а одновременно эффективно определить все допустимые над ними операции. Такие определения типов данных и допустимых операций над ними называются кластерами или абстрактными типами данных (АТД).

В настоящее время преждевременно давать окончательную оценку этому направлению: в мире нет промышленных трансляторов и опыта эксплуатации языков CLU и АЛЬФАРД.

Из огромного многообразия различных технологий программирования следует выделить три направления, безусловно полезных и целесообразных для повсеместного применения:

структурное программирование;

HIPO-технология;

технология главного программиста.

В структурном программировании действия программиста несколько ограничены (некоторая дисциплина проектирования и написания программ): программист может использовать для записи алгоритма только определенные структуры — операторы. Это перераспределяет трудности программирования — проектировать программы становится труднее и оно искусственно затягивается, но зато отлаживать программы становится проще и быстрее, общее число символов в программе уменьшается. Побочным эффектом являются наглядность и читаемость программы. Это позволяет, например, осуществлять передачу разработки программы от одного программиста другому, эффективно модернизировать программу в процессе ее эксплуатации и т. д.

HIPO-технология программирования — это некоторая дисциплина проектирования программ с помощью иерархических диаграмм входа — процесса — выхода (Hierarchical Input — Processing — Output). HIPO-диаграммы — это прием проектирования всей программной системы до записи ее на каком-либо формализованном языке (Фортран, ПЛ/1 и др.). Важным новым элементом HIPO-диаграмм по сравнению, например, с блок-схемами являются выделение данных (на входе и выходе) и запись соответствующего алгоритма в тесной связи с выделенными данными как некоторого процесса преобразования данных на входе в данные на выходе. Такие описания стандартизованы по форме, внешнему представлению HIPO-диаграмм, но язык заполнения диаграмм остается неформальным, как и язык заполнения блок-схем. При этом достигаются две основные цели: качественное и своевременное документирование программного продукта; сокращение времени отладки, особенно комплексной, благодаря тщательности проектирования программного проекта (до его программирования).

Технология главного программиста регламентирует некоторые отношения в коллективе программистов при изготовлении большой системы программ. В целом эти отношения не отличаются от организации любых других коллективов исполнителей и основаны на

дисциплинарном подчинении сверху вниз и специализации программистов на отдельных видах работ: подготовке инструментальных средств, ведении архивов, изготовлении проверочных тестов, документировании программного продукта и др.

Следует заметить, что все отмеченные технологии не подменяют, а взаимодополняют друг друга и описанную ниже Р-технологию программирования; все эти технологии «навязывают» программисту некоторую дисциплину работы. Эта дисциплина, как правило, препятствует выходу программиста немедленно на машину до того, как будет проведено тщательное проектирование алгоритма и согласованы друг с другом взаимодействующие части системы.

Системы построения трансляторов (СПТ) как направление автоматизации программирования имеют два аспекта — теоретический и практический. С теоретической точки зрения это направление весьма перспективно, так как позволяет непрофессиональному программисту сгенерировать транслятор своего языка, задав его формальное описание в некотором метаязыке СПТ. Так как язык занимает важное место в работе прикладных программистов, то СПТ в общем формирует некоторую технологию программирования достаточно массового применения. Особенно велика роль этого направления для научных исследований, так как открывает безграничное поле деятельности по метаязыкам СПТ.

Однако с практической точки зрения в мире нет ни одной промышленной СПТ, имеющей широкое и эффективное применение. Метаязыки существующих СПТ слишком сложны для непрофессиональных программистов. Получаемые с помощью СПТ трансляторы не удовлетворяют требованиям эффективности для промышленной эксплуатации. Из наиболее удачных известных СПТ следует отметить систему CDL (Западный Берлин) и систему ЯРМО (Новосибирск).

Р-технология программирования была разработана в 1971—1976 гг. в Институте кибернетики АН Украинской ССР на основании обобщения большого опыта изготовления и внедрения ряда специальных программных систем*. Р-технология не отвергает и органически включает все лучшее, что есть в описанных выше технологиях и системах программирования. Однако она основана на принципиально новом подходе к решению проблем программирования. Р-технология основывается на переопределении принципов обработки информации в самой ЭВМ таким образом, чтобы учесть и уменьшить специфические отрицательные особенности труда программистов на существующих ЭВМ**. Для этого в Р-технологии программисту предоставляется некоторая программным образом

---

* Буква Р в названии технологии не имеет специальной расшифровки и является продолжением ранее введенных рабочих обозначений.

** Диалектическая необходимость такого переопределения в настоящее время обусловлена характером обрабатываемой на ЭВМ информации — резко возросли объем и сложность этой информации при одновременном упрощении алгоритмов ее обработки (по сравнению с вычислительными алгоритмами первых ЭВМ).

организованная новая вычислительная машина, названная Р-машиной или сокращенно РВМ.

Главный недостаток существующих ЭВМ, который усложняет процесс программирования современных систем обработки информации, заключается в том, что между данными и программой ЭВМ нет прямой связи. Программе ЭВМ, если она не построена специальным образом, безразлично, какие данные обрабатывать. Это означает, что вся сложность связей между данными, программой и абстрактной моделью, которой пользуется и создает программист в процессе проектирования своей программы, ложится на человека. Если объем этих связей превысит некоторый естественный и очень небольшой для человека порог, то начинаются трудности, стоимость преодоления которых очень высока. Из этого недостатка вытекают и трудности организации программистских коллективов, и трудности взаимозаменяемости программистов, слабая специализация труда, почти непреодолимые трудности выпуска хорошей документации для пользователя и др.

Р-технология предлагает:

включить в программу явно заданную связь с данными;

удалить из программы детали, затрудняющие суть обработки информации, которыми являются все операторы ветвления, оставив только линейные операторы типа присваивания выражений, ассоциативного поиска, чтения, записи, обращения к процедурам и функциям и др.;

использовать для записи программ самый простой (и самый древний) язык — язык нагруженных ориентированных графов, которыми задавать как данные, так и последовательности выполнения линейных операторов их обработки.

На дугах графа сверху записывается информация о данных (условие прохождения по дуге), а снизу — соответствующие линейные операторы обработки данных. Если запись сверху отсутствует, то соответствующий оператор выполняется безусловно и затем осуществляется переход по дуге. Такой граф является программой работы Р-машины или Р-программой. Совмещенное двумерное представление информации и алгоритма ее обработки дает следующее.

Во-первых, и прежде всего наглядность программного продукта, которая сохраняется на всех этапах работ по Р-технологии.

Во-вторых, такое представление алгоритма по-новому решает проблему разработки программ. Процесс проектирования Р-программ начинается с формального определения входной информации независимо от какого-либо алгоритма ее обработки (прежде чем что-то обрабатывать, формально определяется, что обрабатывается, в отличие от того, как это делается для ЭВМ: сразу проектируется алгоритм обработки «того — не знаю чего»). Входная информация определяется графом, на дугах которого записываются элементы самой информации.

Затем полученный граф доопределяется до соответствующего алгоритма обработки информации с помощью операторов, записы-

ваемых на дугах графа. Доопределение и распараллеливание работ между исполнителями по определению исходного графа осуществляются по сетевому графику, которым служит сам же граф исходной информации. Работа всех членов коллектива осуществляется единообразно как бы по некоторому конвейеру. При такой организации работ впервые появляется возможность создать технологию промышленного изготовления программ как некоторую производственную линию.

В-третьих, такая запись алгоритма позволяет по-новому организовать отладку программ. Суть заключается в том, что Р-программы обладают одним замечательным свойством, которым принципиально не обладают программы ЭВМ. Р-программы могут работать в специальном инверсном режиме. В данном режиме они не обрабатывают, а, наоборот, генерируют ту информацию, которая может быть на их входе. В какой бы режим не включали программу обычной ЭВМ, мы никогда не получим такой информации. Эта особенность использована в Р-технологии для отладки программ. Специально обученные для этого люди строят контрольно-отладочные Р-программы, включаемые в режим автоматической генерации как угодно большого числа контрольных примеров. Эти примеры автоматически пропускаются через рабочую (отлаживаемую) Р-программу. Диагностика прохождения и сортировка правильных (неправильных) контрольных примеров также происходят автоматически на специальном контрольно-отладочном стенде.

Таким образом, Р-технологию программирования можно назвать технологией двумерного программирования. Выражаясь языком структурного программирования, Р-технологию можно назвать технологией программирования на графических структурах данных только линейными операторами — без операторов перехода, выбора, цикла. Это имеет принципиальное значение для превращения программирования в четко регламентированный производственный процесс, выполняемый коллективом программистов. Следует подчеркнуть, что графическая запись Р-программ особенно перспективна для применения будущих средств вычислительной техники: графических дисплеев, голографических процессоров и т. д. Принципиально новые возможности программирования открываются при аппаратной реализации Р-машины.

В настоящее время для работы по Р-технологии на основных отечественных машинах ЕС ЭВМ и БЭСМ-6 изготовлены универсальные инструментальные средства, названные технологическими комплексами программиста РТК и настроенные на все версии операционных систем на этих машинах (ОС, ДОС, ДУБНА, ДИАПАК). Имеется полный комплект пользовательской документации на комплексы РТК.

Технологические комплексы РТК широко используются для построения различных систем.

Опыт использования РТК в различных организациях показал его эффективность для массового применения. Комплексы РТК облегчают и ускоряют получение готового программного продукта

минимум в 2—3 раза. Максимальная производительность, достигнутая на комплексах РТК БЭСМ-6, составляет 2 тыс. команд на человека в месяц (на отрезке от постановки задачи и согласования ТЗ до внедрения готового и документированного продукта у заказчика). И это не является пределом.

Комплексы РТК осваиваются в среднем за 1,5—2 месяца.

Пользователи, впервые осваивающие Р-технологию и комплексы РТК, отмечают следующие достоинства:

коллективный характер труда программистов по четкой схеме сверху вниз;

возможность работы на РТК непрофессиональных программистов;

устойчивость технологии к ошибкам проектирования, легкость внесения корректив в первоначально принятые архитектурные решения;

возможность эффективного управления сверху ходом выполнения работ, безболезненное подключение новых исполнителей, эффективное распараллеливание работ и организация производственных отношений в коллективе;

очень удобные операторы, предложенные в РТК в качестве стандартных для доопределения информационных графов;

небольшой и посильный для освоения объем документации (по сравнению с документацией на ЕС ЭВМ).

Р-технология хороша там, где задача сведена к задаче преобразования текстов на входе некоторого алгоритма. Желательно, чтобы текст на входе представлялся в виде последовательности символов. Чем сложнее и запутаннее структура допустимых текстов на входе и чем больше их объем, тем лучше для применения Р-технологии.

В связи с этим рассмотрим особенности применения Р-технологии и комплексов РТК для организации и ведения банков данных и пакетов прикладных программ (ППП).

Проблема организации и ведения банков данных и ППП — это в конечном итоге проблема создания языка пользователя, обеспечивающего простой интерфейс с соответствующим программным обеспечением. Проблема такого языка стоит тем более остро при использовании программного обеспечения, адаптированного из-за рубежа или не предназначенного прямо для конкретной области применения. Всегда в любой области применений есть свой профессиональный язык, на котором желательно организовать обращение к универсальному банку данных или пакету программ. С течением времени этот язык может меняться и эволюционировать вместе с изменяющейся средой и опытом пользователей. В этих условиях комплекс РТК является универсальной инструментальной системой, с помощью которой сам пользователь создает нужный ему язык и обеспечивает его эволюцию в процессе эксплуатации. Концептуальная простота Р-технологии и комплексов РТК обеспечивает перспективность такой организации работ.

Есть несколько интересных следствий такого применения РТК, которые могут быть использованы в современных разработках:

с помощью РТК делается лингвистический пакет, который дает возможность пользователю работать на естественном языке;

создается генератор трансляторов для непрофессиональных пользователей и система, обеспечивающая полуавтоматическую настройку (адаптацию) пакета или банка данных на входной язык пользователя;

в качестве универсального входного языка пользователя может быть использован язык графов РТК, расширенный нужными пользователю линейными операторами;

с помощью РТК непосредственно на площадке пользователя может быть сделана сборка (монтаж) пакета программ нужной пользователю комплектации и входным языком. Сборка, обучение пользователя и эксплуатация таких пакетов могут осуществляться централизованно; кроме того, делается универсальный интерфейс между языками программирования широкого применения и системами управления базами данных;

аппаратно реализованными комплексами РТК достигается совместимость пакетов и банков данных, реализованных на различных машинах, что обеспечивает организацию сетей ЭВМ.

Комплекс РТК изготовлен также по Р-технологии. Это дает возможность пользователю, освоившему Р-технологию, целенаправленно модифицировать существующую версию РТК до необходимого на практике уровня. То, что инструментальные средства и проектируемая на них система изготавливаются по единой технологии, имеет неоспоримые преимущества для профессионального роста разработчиков соответствующей системы.

Р-технология и поддерживающее ее программное обеспечение — технологические комплексы РТК являются оригинальными отечественными разработками, в основе которых отсутствует какой-либо зарубежный аналог или прототип.

## ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ

В настоящее время определились следующие основные направления применения вычислительной техники и средств автоматизации:

в экономико-организационных системах (АСУП, ОАСУ);

в системах автоматизации сложных технологических процессов, агрегатов и оборудования (АСУТП);

в системах автоматизации проектно-конструкторских работ (САПР);

в системах обработки данных научных экспериментов и объектов новой техники (АСНИ);

в системах научно-технической информации и информационно-поисковых системах.

Как уже указывалось, основной задачей в экономико-организа-

ционных системах являются максимальное использование оптимизационных расчетов, переход к интегрированию ОАСУ с АСУП, имея в виду создание ведомственной вычислительной сети.

Другая ступень интеграции заключается в тесном взаимодействии АСУП с АСУТП, когда информация с АСУТП вводится в АСУП, вычислительный центр которой выдает в определенном темпе управляющие воздействия в АСУТП.

Следует особо подчеркнуть, что нельзя противопоставлять дальнейшее развитие экономико-организационных систем с работами по АСУТП — они должны развиваться гармонично. Вместе с тем в настоящее время в стране АСУТП примерно вдвое меньше, чем АСУП и ОАСУ, хотя около 8 тыс. технологических процессов должны быть автоматизированы в первую очередь.

Автоматизация технологических процессов производства должна развиваться быстрыми темпами, охватывая все новые и новые отрасли народного хозяйства. Дело в том, что экономическая эффективность АСУТП очень высока благодаря увеличению производительности оборудования на 1—5% за счет более равномерной его работы; повышению выхода годного продукта (материала) вследствие более точного регулирования процесса производства; снижению расхода исходных материалов, топлива, энергии; сокращению численности обслуживающего персонала (сокращения операторов и др.).

В настоящее время, когда в целом по народному хозяйству и отдельным отраслям накоплен большой опыт по созданию и эксплуатации разнообразных АСУТП, имеется полная возможность тиражирования систем с целью использования их для автоматизации аналогичных технологических процессов.

Принципы типизации АСУТП заложены в утвержденных Госкомитетом по науке и технике Общеотраслевых руководящих методических материалах (ОРММ) по созданию АСУТП.

Примером применения типового проектирования может служить автоматизированная система типа «Карат» для управления горно-транспортными работами в открытых карьерах. Система прошла опытную проверку на пяти предприятиях и обеспечивает повышение на 5—10% использования большегрузных самосвалов и окупается, как правило, менее чем за 0,5 года. Система «Нефелин-1» на Пикалевском глиноземном комбинате им. 50-летия СССР, созданная на базе двух ЭВМ М-6000, осуществляет оптимальное управление технологическими процессами приготовления нефелино-известняковой шихты, производства глинозема, соды, портландцемента и других продуктов. Применение этой системы позволило увеличить выпуск глинозема на 1% и сократить обслуживающий персонал на 60 человек. Затраты на создание системы составили на январь 1978 г. 3,2 млн. руб., а срок их окупаемости — 1,8 года.

АСУТП переработки нефелина имеет большое значение при необходимости точного поддержания параметров процесса. Это требование может быть выполнено для регулирования управляющих ЭВМ. Опыт Пикалевского глиноземного комбината необходимо ис-

пользовать как на Ачинском глиноземном комбинате, так и на других аналогичных предприятиях.

АСУТП кислородно-конверторного цеха на Западно-Сибирском металлургическом заводе, построенная на базе четырех ЭВМ М-6000, управляет работой двух конверторов емкостью по 350 т. Система обеспечивает проведение расчета состава шихты, расхода кислорода, сыпучих добавок и времени продувки, выдачу рапортов о работе оборудования и смен, проведение автоматической обработки данных химического анализа чугуна и стали. В результате число плавок, выпускаемых с первой повалки конвертора, увеличилось на 20%, что дает повышение производительности конвертора на 5—7%; благодаря стабилизации температурного режима улучшилась (на 13%) стойкость футеровки конвертора; выход годного металла повысился на 0,5%. Годовой экономический эффект от применения этой системы составляет 850 тыс. руб.

Автоматизация проектирования и конструирования является эффективным направлением использования вычислительной техники.

Отечественный и зарубежный опыт показывают, что применение математических методов и ЭВМ при проектировании, конструировании и в научных исследованиях повышает технический уровень и качество проектируемых объектов, сокращает сроки разработки и освоение их производства. Автоматизация процессов проектирования особенно эффективна после перехода от автоматизации отдельных инженерных расчетов к созданию систем автоматизированного проектирования и конструирования, охватывающих все стадии этих работ — от проектирования до технологической подготовки производства. Экономический эффект получается за счет существенного ускорения разработки и повышения качества проектируемых объектов на основе принятия оптимальных решений, в результате чего достигаются также уменьшение материалоемкости в машиностроении, экономия материалов и капитальных затрат при строительстве промышленных объектов, сооружений трубопроводов,.дорог и т. д.

Создание САПР в X пятилетке предусмотрено программами работ, утвержденными ГКНТ, а также планами министерств и ведомств. В текущей пятилетке САПР создаются более чем в 50 организациях промышленности и строительства.

Применение САПР приводит к ускорению исследований и разработок в 1,5—2 раза; при использовании САПР на 5—10% уменьшается материалоемкость вновь проектируемых изделий и объектов, на 10—15% уменьшаются расходы на строительство и эксплуатацию оптимально спроектированных объектов. Например, использование ЭВМ при проектировании объектов Федоровского, Тепловского и Савуйского месторождений нефти в Западной Сибири позволили сократить объемы капитальных вложений на 2,6 млн. руб. за счет выбора оптимальных вариантов генеральных схем обустройства этих месторождений. При проектировании Оскольского электрометаллургического комбината удалось сэкономить за счет

оптимизации решения генплана 6,4 млн. руб. при одновременной экономии 10—12% материалов и конструкций.

Для проведения единой технической политики по созданию САПР в проектных, конструкторских и технологических организациях ГКНТ разработаны и в 1978 г. утверждены «Общеотраслевые руководящие методические материалы по созданию САПР». В этом документе обобщен опыт создания САПР и установлены основные принципы создания, эксплуатации и развития систем автоматизированного проектирования.

Автоматизация проектных конструкторских работ должна развиваться в двух основных направлениях: создание САПР примерно в 50% крупных проектно-конструкторских организациях, в промышленности и строительстве на базе индивидуальных вычислительных центров; широкое применение в остальных проектно-конструкторских организациях типовых программ и инженерных расчетов на ЭВМ, входящих в ГСВЦ, с помощью терминальных устройств абонентских пунктов.

Для создания указанных САПР в соответствующих проектно-конструкторских организациях потребуются мощные ЭВМ с быстродействием до 10 и более миллионов операций в секунду и большое число малых, мини- и микро-ЭВМ. Для систем САПР разрабатывается и будет выпускаться широкая номенклатура графопостроителей, дисплеев, систем ввода в ЭВМ информации с микрофильмов и вывода информации на микрофильмы и стеклографические репродукционные системы и др.

Применение вычислительной техники повышает эффективность деятельности научно-исследовательских организаций. Наибольший эффект получается при создании автоматизированных систем для обработки данных научных исследований, управления экспериментами и комплексными испытаниями объектов новой техники и технологии (АСНИ). В текущей пятилетке АСНИ создаются более чем в 40 организациях АН СССР, Минвуза СССР и промышленных министерств. С целью сокращения капитальных вложений в АСНИ должен широко применяться принцип коллективного использования этих систем.

Для АСНИ будут использованы ЭВМ всех классов, в том числе малые и микро-ЭВМ. Для создания систем потребуется большое число специализированной аппаратуры и значительное количество различных датчиков.

Особое внимание будет уделено организации серийного производства стандартных систем сбора информации и управлению объектами исследования на основе типовых блоков системы «Камак». В частности, необходимо решить проблему оснащения АСНИ высокоточными устройствами ввода-вывода графической информации и ее регистрации.

Нельзя не упомянуть о системах подготовки данных. Серьезным недостатком комплектации АСУ и ВЦ является система подготовки данных, основанная на бумажных носителях (перфо-

картах и перфолентах) и устаревших типах оборудования для их обработки.

В настоящее время в большинстве ВЦ и АСУ используются морально устаревшие устройства подготовки информации на перфокартах. Перфокарточное оборудование имеет низкую производительность, требует большого количества ручных операций, носитель может использоваться только однократно и для хранения требует много места. Существенным недостатком перфорационных устройств является большой расход высококачественной бумаги.

По оценочным данным потребность в перфокартах для работы ЭВМ составит в текущей пятилетке около 30 млрд. штук, для изготовления перфокарт потребуется 80 тыс. т специальной бумаги.

В последние годы широко внедряются экономически более эффективные средства подготовки данных на магнитных носителях информации. Такие средства имеют ряд преимуществ по сравнению с перфорационными: стоимость подготовки данных на 30—50% ниже, автоматизируется контроль вводимой информации, производительность труда операторов повышается на 50—60%, более чем в 10 раз увеличивается скорость ввода информации в ЭВМ, для хранения требуется меньше места (одна бобина эквивалентная 32 тыс. перфокарт), стоимость хранения информации снижается в 8—10 раз.

Магнитная лента в отличие от перфокарт и перфолент может использоваться многократно и поэтому резко снижается расход носителей информации при подготовке данных.

В нашей стране разработаны и серийно выпускаются устройства подготовки данных на магнитной ленте ЕС-9001 и СПД-9000 (на базе мини-ЭВМ М-6000). Расчет экономической эффективности применения одной системы СПД-9000, проведенный ВНИПИОАСУ (система СПД-9000 заменяет 1 табулятор, 2 сортировки, 14 перфораторов и 34 суммирующие машины), показывает, что общая годовая эффективность от использования одной системы составляет 37,4 тыс. руб. и при ее применении экономится 5 млн. перфокарт в год.

Объем выпуска устройств ЕС-9001 и СПД-9000 совершенно недостаточен для удовлетворения потребностей народного хозяйства, по техническому уровню указанная аппаратура уступает современным зарубежным устройствам аналогичного назначения.

В заключение следует сказать, что в статье отмечены далеко не все проблемы и нерешенные задачи в области разработки, производства и использования вычислительной техники.

Задачи ученых, инженеров и всех, кто разрабатывает, производит и эксплуатирует вычислительную технику, заключается в том, чтобы полностью удовлетворить все отрасли народного хозяйства в информационно-вычислительных работах, ускоренными темпами повышать эффективность общественного производства.

*В. М. САВИНКОВ, М. С. КАЗАРОВ,*
*Ю. К. РЫСЕВИЧ*

# ИСПОЛЬЗОВАНИЕ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ В АСУ*
## (КРАТКИЙ ОБЗОР)

Применение в автоматизированных системах управления технических (в первую очередь запоминающих устройств прямого доступа) и программных средств ЭВМ третьего поколения привело к появлению новых технологий обработки данных, направленных на дальнейшее повышение эффективности и качества функционирования АСУ. В связи с этим прежде всего следует отметить разработку и реализацию принципиально новых подходов к созданию и использованию информационной базы АСУ, т. е. реализацию концепции баз данных, позволяющей осуществить интегрированную организацию и хранение данных для многопланового применения в различных приложениях.

За рубежом разработка баз данных общего назначения остается одним из основных направлений деятельности в области обработки данных. Ожидается, что через 10—15 лет индустрия обработки данных превратится в одну из основных отраслей промышленности США, а в настоящее время около 20% объема национального продукта Соединенных Штатов расходуется на накопление, обработку и распространение знаний. Стоимость хранения и поддержания одного байта информации в настоящее время доведена до стоимости одного полиграфического знака, при значительно больших возможностях последующего применения и многократного автоматического использования. Практически каждая десятая вычислительная установка в США в 1977 г. имела систему управления базами данных (СУБД) в качестве основы для построения информационной системы.

На основании анкетного опроса пользователей СУБД было отобрано девять систем, получивших наибольшее одобрение. Наивысшую оценку получила система Ingnire, далее следует IDMS, ADABAS, TOTAL, System 2000, IMS, DBOMP, DL/1, GIS.

В нашей стране интенсивно развивается разработка банков данных в различных направлениях. Основное направление применения банков данных — автоматизированные системы управления технологическими процессами, предприятиями, объединениями, отраслями и автоматизация проектирования. Из межотраслевых систем, в рамках которых создаются банки данных, следует выделить экономические, в том числе и статистические, социально-политические банки данных, банки научно-технической информации. Особое место занимает создание банков данных вычислительных центров кол-

* Статья написана на основе материалов Временной научно-технической комиссии по техническому уровню разработок и внедрению в автоматизированные системы управления комплексов программ для организации и ведения на ЭВМ 3-го поколения банков данных.

лективного пользования (ВЦКП), где ожидается наибольшая эффективность от реализации концепции баз данных.

При переходе к системам баз данных сокращаются затраты на обслуживание данных благодаря устранению их избыточности при хранении и появляется возможность контролируемого доступа к данным с целью обеспечения защиты от разрушения или несанкционированного использования, что является немаловажным для повышения эффективности, надежности и качества функционирования человеко-машинных систем, какими являются АСУ.

Другими важными преимуществами использования концепции баз данных являются:

возможность легкого доступа пользователей к требуемым данным, гибкость в обращении с данными благодаря различным методам доступа, высокая производительность, быстрая обработка случайных запросов на данные, простота внесения изменений, наличие интерфейса высокого уровня для связи с программистом;

возможность взаимодействия пользователя с системой на непроцедурном языке запросов, возможность автоматически восстанавливать данные после разрушения, осуществлять автоматическую реорганизацию данных и др.

Особенно важным является реализация принципа логической и физической независимости данных. Как известно, современные АСУ представляют собой постоянно развивающиеся и совершенствующиеся системы. Совершенствование старых и включение в систему новых прикладных программ не могут не оказывать влияния на структуру и состав базы данных. Суть принципа независимости прикладных программ от данных состоит в том, что изменение данных, используемых одной прикладной программой, не оказывает влияния на другие прикладные программы. В случае реконструкции логического описания данных прикладные программы при этом не перезаписываются. Изменения физической организации данных и аппаратных средств также не оказывают влияния на прикладные программы. Отображение представлений данных на логический и физический уровни выполняется СУБД.

### Состояние работ по СУБД

В нашей стране первые практические результаты по СУБД начали появляться с 1975 г., когда широкое распространение получили ЭВМ третьего поколения. Отсутствие у разработчиков опыта работы с внешней памятью прямого доступа, потребность в использовании СУБД привели к тому, что первые СУБД появились как результат адаптации наиболее распространенных СУБД за рубежом. Наибольшую известность в настоящее время получили следующие системы:

СМО «Банк данных АСУП» (СИОД) версий СИОД-1, СИОД-2, СИОД-ОС;

секторно-ориентированная система программирования версий НСИ-ДОС, НСИ-1-ДОС;

СМО «Банк данных универсальной структуры» версий «БАНК», «БАНК-ОС»;

система ведения банков данных иерархической структуры, СИНБАД-2;

система управления базами данных «Ока»;

операционная система телеобработки «Кама»;

система управления базой данных «Набоб»;

информационная экономическая система ИНЭС-2;

система организации баз данных «Пальма».

Общие сведения, операционная обстановка и характеристики рассмотренных СУБД приведены в табл. 1, 2.

Работы в области проектирования СУБД в настоящее время продолжаются.

В ЦЭМИ АН СССР ведется разработка средств организации и ведения экономических банков данных, НИИ ЦСУ СССР разрабатывает средства ведения банка данных АСГС. Начаты работы по созданию средств организации и ведения распределенных банков данных.

### Ориентация применения СУБД

Несмотря на общее назначение, различные СУБД существенно отличаются друг от друга по возможностям, предоставляемым пользователям, принципам организации, операционной обстановке и т. п.

Системы управления базами данных семейства СИОД и НСИ-ДОС предназначены для создания и обслуживания баз данных АСУ промышленных предприятий с дискретным характером производства, таких, как машиностроительные и приборостроительные. Важным свойством этих систем является наличие специальных средств, обеспечивающих построение программ, эффективно реализующих процедуры типа разузлования (для получения информации о составе и применимости изделий). Другая важная особенность этих систем заключается в том, что пользователь получает возможность применения ряда пакетов прикладных программ функционального назначения, ориентированных на применение в АСУ промышленными предприятиями. Кроме того, СУБД НСИ-ДОС может использоваться при организации нормативно-справочной информации АСУ при решении задач подсистемы технической подготовки производства и с некоторыми ограничениями в АСУ непромышленной сферы. В качестве примера можно привести использование СУБД СИОД-2 в Петрозаводском машиностроительном производственном объединении имени Ленина для решения задач суммарного и структурного разузлования, расчета спецификаций и расходов материалов на изделие и по цехам, маршрутизации и составления ведомости материалов.

В настоящее время нет достаточного опыта эксплуатации СУБД «Ока» и СИНБАД-2. Однако по мнению разработчиков они

| Системы | Основная сфера использования | ППП, использующие СУБД | Число эксплуатируемых систем | |
|---|---|---|---|---|
| | | | всего | в том числе с сопровождением |
| СИОД-2 | АСУП предприятий с дискретным характером производства | Планирование потребности<br>Планирование мощности<br>Управление цехом<br>Управление запасами | 335 | 39 |
| СИОД-ОС | То же | Планирование потребностей | — | — |
| НСИ-ДОС | » | Подсистемы оперативного управления производством | 31 | 31 |
| СИНБАД-2 | Универсальная, широкого назначения | — | 150 | 13 |
| «Ока» | То же | — | 25 | 25 |
| «Кама» | Универсальная, с ориентацией на обеспечение теледоступа | — | 25 | 25 |
| «БАНК» | Универсальная, ориентация на АСУП, АСУ непромышленных объектов | Информационная база АСУ «Тверь-2» | 463 | 98 |
| «БАНК-ОС» | То же | — | 35 | 2 |
| «Набоб» | Универсальная, широкого назначения | — | — | — |
| ИНЭС-2 | Экономическая, широкого назначения | — | — | — |
| «Пальма» | Универсальная, широкого назначения, ориентация на АСУ | — | 5 | 5 |

могут быть предназначены для любой сферы управления народным хозяйством: АСУП, ОАСУ, ИПС и т. п. Ожидается, что наибольший эффект эти системы дадут при использовании в больших системах обработки данных, характеризующихся:

постоянным расширением типов обрабатываемых данных;

высокими требованиями к надежности;

Таблица 1

дения о СУБД

| Минимальная конфигурация ЭВМ | Рекомендуемая конфигурация ЭВМ | Операционная система, версия | Количество команд | Обучение и консультация пользователей |
|---|---|---|---|---|
| 32К оп. памяти, 2 НМД, 1 НМЛ | 64К оп. памяти, 2 НМД, 1 НМЛ | ДОС ЕС, начиная 1.3 и выше | 40,3 тыс. маш. ком. | НПО «Центрпрограммсистем», ЛИМТУ, Ленинград |
| 128К оп. памяти, 2 НМД | 256К оп. памяти, 2 НМД | ОС ЕС | — | НПО «Центрпрограммсистем», г. Калинин |
| 32К оп. памяти, 2 НМД | 64К оп. памяти, 3 НМД | ДОС ЕС любой версии | 28,7 тыс. маш. ком. | НПО «Центрпрограммсистем», г. Калинин |
| 128К оп. памяти, 3 НМД | 256К оп. памяти, 3 НМД, 2 НМЛ | ОС ЕС (MFT, MYT) | 64,4 тыс. маш. ком. | НПО «Центрпрограммсистем», г. Калинин; ЛИМТУ, Ленинград |
| 128К оп. памяти, 3 НМД, 1 НМЛ | 512К оп. памяти, 4 НМД, 2 НМЛ | ОС ЕС (MFT, MYT) 4.0 и выше | 100 тыс. маш. ком. | ИК АН УССР, г. Киев |
| 256К оп. памяти, 4 НМД, 1 НМЛ | 512К оп. памяти, 4 НМД, 1 НМЛ | ОС ЕС 4.0 и выше | 50 тыс. маш. ком. | ИК АН УССР, г. Киев |
| 64К оп. памяти, 2 НМД | 128К оп. памяти, 2 НМД | ДОС ЕС 1.3 и выше | 5,9 тыс. маш. ком. | НПО «Центрпрограммсистем», г. Калинин; ЛИМТУ, Ленинград |
| 128К оп. памяти, 3 НМД | 256К оп. памяти, 3 НМД, 2 НМЛ | ОС ЕС 4.0 и выше | 10 тыс. маш. ком. | НПО «Центрпрограммсистем», г. Калинин; ЛИМТУ, Ленинград |
| 128К оп. памяти, 2 НМД | 256К оп. памяти, 3 НМД | ДОС ЕС 2.0 и выше | 300 Кбайт | — |
| 256К оп. памяти, 3 НМД, | 512К оп. памяти, 4 НМД | ОС ЕС 4.0 и выше | — | — |
| 128К оп. памяти, 2 НМД, 1 НМЛ | 256 оп. памяти, 3 НМД, 2 НМЛ | ДОС ЕС 2.1 и выше ДОС ЕС 1.3 | 120 Кбайт | ИК АН УССР, г. Киев |

необходимостью централизации управления данными;

дальнейшим переходом на применение средств телеобработки.

Работы по созданию банка данных с использованием системы «Ока» ведутся в ГВЦ Госплана СССР, в ряде АСУ машиностроительными отраслями, в АСУ «Главтюменьнефтегаза» и др. СУБД СИНБАД-2 в настоящее время внедряется в автоматизированную

Функциональные характеристики СУБД

Таблица 2

| | СИОД-2 | СИОД-ОС | НСИ-ДОС | СИНБАД-2 |
|---|---|---|---|---|
| Класс СУБД | Б, З | Б, З | Б | Б |
| Тип конечного пользователя | ПП, НП | ПП, НП | ПП | ПП |
| Режимы функционирования | Пакетный | Пакетный | Пакетный | Пакетный |
| Допустимые логические структуры данных | Фиксированные сети | Фиксированные сети | Фиксированные сети | Иерархическая (ПП), огранич. сеть (АД) |
| Используемые методы доступа | Комбинации индексно-последовательного и прямого | Комбинации индексно-последовательного и прямого | Последовательный, индексно-последовательный | Иерархический последовательный, иерархический индексно-последовательный, иерархический прямой, иерархический индексно-прямой |
| Языковые средства: | | | | |
| администратора базы данных | Параметры генерации схемы | Параметры генерации схемы | Параметрический ЯОД | Язык описания структур базы данных |
| прикладного программиста | Макрокоманды, включаемые в Ассемблер, ПЛ/1, Кобол | Макрокомады, включаемые в Ассемблер, ПЛ/1, Кобол | Макрокоманды, включаемые в Ассемблер | Языки оперирования данными, включаемые в Ассемблер, Кобол, ПЛ/1 |
| непрограммиста | Параметрический язык генерации поисковых процедур | Параметрический язык генерации поисковых процедур | Нет | Нет |
| Обеспечение независимости данных (число уровней логического описания) | Один | Один | Один | Два |
| Средства защиты данных: | | | | |
| от несанкционированного доступа | Нет | Нет | По кодам памяти и доступа | Есть |
| физической целостности | Проверка значений по порц. сравн. | Проверка знач. по порц. сравн. | Проверка значен. по порц. сравн. | Широкие возможности |
| Средства ведения журналов и организации рестарта | Нет | Нет | Нет | Есть |
| Средства реорганизации | Есть | Нет | Есть | Есть |
| Сбор и обработка статистики | Нет | Нет | Нет | Есть |

*Продолжение табл. 2*

| | «Ока» | «Кама» | «БАНК» | «БАНК-ОС» |
|---|---|---|---|---|
| Класс СУБД | Б | Б | Б | Б |
| Тип конечного пользователя | ПП | ПП | ПП | ПП |
| Режимы функционирования | Пакетный, теледоступ | Теледоступ, пакет. | Пакетный | Пакетный |
| Допустимые логические структуры данных | Иерархическая (ПП), огранич. сеть (АД) | Фиксированные сети | Сети | Сети |
| Используемые методы доступа | Иерархический последовательный, иерархический индексно-последовательный, иерархический прямой, иерархический индексно-прямой | Последовательный, индексно-последовательный, прямой, индексно-прямой, косвенный | Прямой по ключу, прямой по адресу, относительно главной записи | Прямой по ключу, прямой по адресу, относительно главной записи |
| Языковые средства: | | | | |
| администратора базы данных | Язык описания структур базы данных | Параметрический ЯОД | Декларативные макрокоманды | Декларативные макрокоманды |
| прикладного программиста | Языки оперирования данными, включаемые в Ассемблер, Кобол, ПЛ/1 | Набор макрокоманд, включаемых в Ассемблер, Кобол, ПЛ/1 | Макрокоманды, включаемые в Ассемблер, Кобол | Макрокоманды, включаемые в Ассемблер |
| непрограммиста | Нет | — | Нет | Нет |
| Обеспечение независимости данных (число уровней логического описания) | Два | Один | Один | Один |
| Средства защиты данных: | | | | |
| от несанкционированного доступа | Есть | Есть | Нет | Нет |
| физической целостности | Широкие возможности | Автономное восстановление | Автономное восстановление | Автономное и автоматическое восстановление |
| Средства ведения журналов и организации рестарта | Есть | Только журнал | Только журнал | Есть |
| Средства реорганизации | Есть | Нет | Есть | Есть |
| Сбор и обработка статистики | Есть | Есть | Есть | Есть |

*Продолжение табл. 2*

| | «Набоб» | ИНЭС-2 | «Пальма» |
|---|---|---|---|
| Класс СУБД | Б | Б, З | Б, З |
| Тип конечного пользователя | ПП | ПП, НП | НП, НП |
| Режимы функционирования | Пакетный | Пакетный, теледоступ | Пакетный |
| Допустимые логические структуры данных | Сети | Смешанные формы | Реляционные |
| Используемые методы доступа | Индексно-последовательный для организации, прямой по ключу для доступа | Собств. метод (динамич. метод) | Прямой, индексно-последовательный для орган., прямой по ключу для поиска |
| Языковые средства: | | | |
| администратор базы данных | ЯОД схемы и подсхемы | Входные языки системы и языки перестройки | ЯОД схемы и подсхемы |
| прикладного программиста | ЯМД, включаемый в ПЛ/1 | Расшир. алгоритм. языков | ЯМД включ. в Ассемблер, ПЛ/1 |
| непрограммиста | Нет | Язык запросов «Доступ» | Язык запросов «Дисплей» |
| Обеспечение независимости данных (число уровней логического описания) | Два | Один | Два |
| Средства защиты данных: | | | |
| от несанкционированного доступа | Замки секретности | Есть | Нет |
| физической целостности | Автономное и автоматическое восстановление | Есть | Есть |
| Средства ведения журналов и организации рестарта | Есть | Есть | Только журнал |
| Средства реорганизации | Нет | Есть | Нет |
| Сбор и обработка статистики | Возможность подключения процедур администратора | Есть | Нет |

**Примечание.** Б — СУБД с базовым языком; З — замкнутая организация; ПП — прикладной программист; НП — непрограммист.

систему управления Всесоюзного объединения государственных цирков.

Пакет «Кама» — это общесистемный пакет прикладных программ, расширяющий возможности ОС ЕС при работе с базой данных в оперативном режиме и обеспечивающий управляемый доступ пользователей с локальных и удаленных терминалов к базе данных и библиотеке прикладных программ, написанных на языках Ассемблер, ПЛ/1 и Кобол.

Для построения информационных систем, работающих в оперативном режиме, «Кама» обеспечивает:

единую базу данных для всех прикладных программ;

управляемый доступ к базе данных;

управление работой телекоммуникационной сети разнородных терминалов;

эффективное управление ресурсами;

приоритетное использование средств обработки;

сервисные средства, необходимые для построения информационных систем. Опытная эксплуатация пакета «Кама» проводится в ГВЦ Госплана СССР, ИК АН УССР и в других организациях.

В настоящее время система «БАНК» применяется для информационного обеспечения автоматизированных систем управления промышленными предприятиями машиностроения, предприятий с серийным и крупносерийным типом производства с подетальной и позаказной системой планирования для решения задач ведения нормативно-справочного хозяйства, задач технической подготовки производства. Кроме того, планируется использование этой системы для создания баз данных управления заказами, нормативной базы, оперативного управления сбытом.

СУБД «БАНК» функционирует:

в «АСУ-Криогенмаш» (номенклатура 180 тыс. деталей, 900 тыс. деталеопераций, 500 тыс. наименований материалов). База данных, занимающая 9 пакетов магнитных дисков, содержит конструкторско-технологические спецификации, трудовые и материальные нормативы;

в АСУ-УВМ (завод управляющих вычислительных машин). Кроме нормативной информации, в базе данных хранятся и оперативные данные;

в АСУ-ГПЗ-1 для ведения конструкторско-технологических спецификаций и трудовых нормативов.

СУБД «БАНК» внедряется также в АСУ хлорной промышленности в подсистемы текущего планирования, оперативного управления и капитального строительства.

Возможности применения «БАНК-ОС» аналогичны.

Система «Набоб» представляет собой универсальную систему управления, отвечающую требованиям различных по назначению информационных систем. Она базируется на предложениях DBTG CODASYL в отношении структур базы данных, языков описания данных и рассчитана на функционирование в среде ДОС ЕС.

ИНЭС-2 — универсальная информационная система предполагается для использования в автоматизированных системах управления: в частности, в автоматизированной системе плановых расчетов (АСПР) Госплана СССР и госпланов союзных республик, в автоматизированной системе государственной статистики (АСГС) ЦСУ СССР, в отраслевых АСУ, а также в научных целях, при создании комплексов обработки данных. Эта разработка является дальнейшим развитием первой версии системы, реализованной на ЭВМ ICL в 1976 г.

## Проблемы использования СУБД

Системы управления базами данных, обеспечивая существенные преимущества для организации информационных систем, в то же время выдвигают свои проблемы.

**Проблема технического обеспечения.** Наличие надежных ЭВМ с эффективным быстродействием, разнообразных средств непосредственного общения абонентов с ЭВМ и достаточно быстрых каналов связи как для ввода информации, так и для поиска информации, наличие специальных автоматизированных средств тиражирования различных документов для избирательного распределения информации. СУБД предъявляет высокие требования к объему внешней памяти прямого доступа. Преимущества СУБД достигаются в полной мере только в том случае, когда объем памяти прямого доступа, одновременно подключаемой к вычислительной системе, достаточен для размещения всей базы данных. Наиболее распространенные в настоящее время дисковые запоминающие устройства емкостью 7,25 Мбайт для средних и крупных систем оказываются малопригодными. С переходом на дисковые запоминающие устройства емкостью 29 Мбайт уже будут достигнуты удовлетворительные результаты, а емкостью 100 Мбайт — проблема будет решена полностью.

**Проблема проектирования баз данных.** Характеристики работы СУБД, в частности время реакции, существенным образом зависят от того, как спроектирована БД (выбраны информационные единицы, определены связи между ними, выбраны методы доступа и т. п.). Обычно проектированию предшествует детальный анализ объекта управления с целью создания наиболее точной и полной информационной модели. Стремление достичь максимальной полноты и точности модели направлено на сокращение больших непроизводительных затрат ресурсов ЭВМ на реорганизацию БД в связи с развитием и совершенствованием системы.

В настоящее время не существует более или менее удовлетворительных подходов к решению всей проблемы проектирования БД. Большие надежды в связи с этим возлагаются на использование автоматизированных систем проектирования баз данных. Работы в этом направлении уже ведутся.

*А. А. КАМЕНЦЕВ*

# ОРГАНИЗАЦИЯ СТРУКТУРЫ ДАННЫХ ПРИ РЕГИСТРОВОЙ ФОРМЕ ХРАНЕНИЯ И ОБРАБОТКИ ИНФОРМАЦИИ

В совершенствовании информации, сокращении ее объемов, улучшении методологии расчета показателей большая роль принадлежит современным методам наблюдения, хранения и обработки статистической информации, базирующимся на широком использовании единых унифицированных форм статистической отчетности, общесоюзных классификаторов и систем обозначений, переводе крупных статистических работ на электронную обработку.

В настоящее время при выполнении какой-либо аналитической работы используется ограниченное число показателей, определяемое числом соответствующих взаимосвязанных форм статистической отчетности. В связи с этим информация, содержащаяся в выходных таблицах, в ряде случаев не обеспечивает проведения глубокого экономического анализа. Для получения качественных результатов возникает потребность в дополнительной информации, хранящейся в других разделах статистики, что влечет за собой увеличение объема обрабатываемых данных.

В устранении отмеченных недостатков большая роль принадлежит вопросам рациональной организации формирования и хранения исходных данных. Статистические данные должны удовлетворять развивающимся потребностям органов управления, поэтому следует учитывать разные аспекты информационного обеспечения, применяемого при решении задач планирования и управления на любых уровнях народного хозяйства.

В этих условиях весьма перспективной представляется идея создания и хранения на соответствующих уровнях АСГС банков данных, содержащих всю разновидность системы показателей, используемых при решении различных задач. Создание универсальных банков данных осложняется в связи с необходимостью ведения широкой номенклатуры показателей, отражающих все разнообразие и специфику функций планирования и учета на разных уровнях управления. Кроме того, использование большого количества данных еще недостаточно для принятия обоснованных решений, поскольку для комплексного анализа требуется их методологическая увязка, возможность проведения многократной перегруппировки данных для решения конкретной задачи. Поэтому в условиях разноаспектной потребности в информации наиболее рациональным принципом формирования банка данных, по-видимому, будет такой, при котором выбор показателей обеспечивает решение статистических задач с учетом комплексности обработки и анализа данных.

Получение таких данных наиболее целесообразно осуществлять регистровой формой наблюдения, при которой хранение информа-

ции организуется из записей однородной совокупности объектов наблюдения. Такая форма наблюдения и хранения информации обеспечивает ведение и обработку данных независимо от структуры статистических документов, позволяя осуществить системную организацию информации по каждому наблюдаемому объекту и комплексную ее обработку по всем направлениям. Эффективность данной формы хранения особенно проявляется при ведении относительно устойчивых совокупностей объектов, имеющих продолжительный период действия и набор общих свойств, существенно не изменяющихся в течение этого времени. Поэтому регистр можно рассматривать как автоматизированную картотеку однородной совокупности единиц статистического наблюдения определенного типа.

Ниже рассматривается общая схема организации формирования структуры и описания данных на примере регистровой формы хранения и обработки информации, характеризующей производственно-хозяйственную деятельность промышленных предприятий. Данная схема описывает объекты наблюдения в виде совокупности трех компонентов: блока идентификации промышленных предприятий, блока системы экономических показателей, блока информационно-справочных данных.

На основании экспериментальных данных, полученных при создании и функционировании регистра промышленных предприятий (РПП) в ЦСУ Эстонской ССР в проектировании регистровой формы хранения достаточно четко определились два подхода к организации данных: а) создание единого массива данных; б) разделение фонда данных на подмассивы или сегменты. Первый из них предполагает последовательный просмотр всего массива данных. Записи в массиве располагаются, как правило, в порядке возрастания (убывания) идентификационных кодов объектов. В целях сокращения времени поиска используются вспомогательные картотеки (таблицы), содержащие сведения о границах расположения определенных групп кодовых номеров в общем массиве данных.

В системе с фондом, разделенном на подмассивы, информация группируется в блоки (группы). Принцип группировки может быть самым различным. Например, внутри каждого из подмассивов информация может располагаться произвольным образом, упорядочиваться по какому-либо признаку или может быть также разбита на подгруппы. В этом случае поиск ограничивается просмотром сравнительно небольшого числа записей. При данном подходе к построению фонда данных расходуется меньше времени на поиск по сравнению с предыдущим. Он особенно эффективен при автономном обращении к отдельным показателям или группе показателей. Данное обстоятельство во многих случаях является определяющим при выборе структуры организации данных, поскольку на практике частота обращений к отдельным показателям, как правило, превалирует над обращениями к полному массиву хранимых данных. Поэтому при частом обращении к отдельным показателям блочная структура является наиболее рациональной, поскольку предусматривает, с одной стороны, разделение основного информационного массива на

подмассивы, формируемые для решения соответствующих классов задач, с другой — позволяет оперативно получать необходимые данные о предприятиях или их совокупностях, сгруппированных по определенному признаку (например, по отраслевой принадлежности).

При такой структуре основной информационный массив представляется в виде следующих подмассивов: подмассив, содержащий идентификатор объекта, являющийся входом в регистр, его наименование, классификационные признаки, выступающие в роли ключей при выполнении отдельных процедур обработки статистической информации; подмассив, содержащий систему экономических показателей с разбивкой на подмассивы-сегменты, соответствующие определенным группам показателей (например, показателям, характеризующим продукцию, основные фонды, численность работающих и т. д.); подмассив, содержащий информационно-справочные данные объекта наблюдения. Такая структура расположения записей представлена ниже:

| Идентификационный номер | Наименование объекта | Классификационные признаки | Показатели 1-й группы | | ... | Показатели $n$-й группы | | Справочные данные |
|---|---|---|---|---|---|---|---|---|
| | | | | продукция | | | фонды | |
| Корневой сегмент | | | 1-й информационный сегмент | | | $n$-й информационный сегмент | | справочный сегмент |

Первые три элемента регистра являются элементами Общесоюзного классификатора предприятий и организаций (ОКПО), обеспечивающими формализованное и однозначное описание объектов, и выступают как корневой сегмент регистра. Показатели, характеризующие производственно-хозяйственную деятельность предприятий, формируются в группы в соответствии с их функциональным назначением, образуя информационные сегменты, а данные информационно-справочного блока образуют справочные сегменты.

Таким образом, полная запись по каждому объекту будет содержать идентификационный номер предприятия и соответствующие группы экономических и справочных показателей. В этом случае идентификаторы объектов выполняют роль связующего звена информационных блоков, а классификационные признаки (коды министерства, отрасли, территории) выступают в роли ключей при выполнении отдельных процедур обработки статистической информации.

Как показали результаты эксперимента, проводимого на базе использования ЭВМ «Минск-32», рассмотренная выше структура регистра рациональна для ограниченного круга описываемых данных, содержащих около 300 объектов и 100 первичных экономиче-

ских показателей. В случае расширения границ регистра данная структура не позволяет эффективно решать задачи, связанные с обработкой большой номенклатуры данных, поскольку, как показывают расчеты, требуется значительное количество машинных носителей информации. В этих условиях резко повышается трудоемкость процедуры поиска и выборки соответствующих показателей или группы показателей.

Для регистров, содержащих большую номенклатуру данных, в качестве основной запоминающей среды целесообразно использовать устройства с прямым доступом — магнитные диски (МД), а в качестве вспомогательной — магнитные ленты. В этом случае создаются предпосылки автономного хранения корневого и информационного сегментов. В качестве связующего звена этих сегментов могут выступать адреса-связки, использование которых наиболее эффективно при применении ЕС ЭВМ.

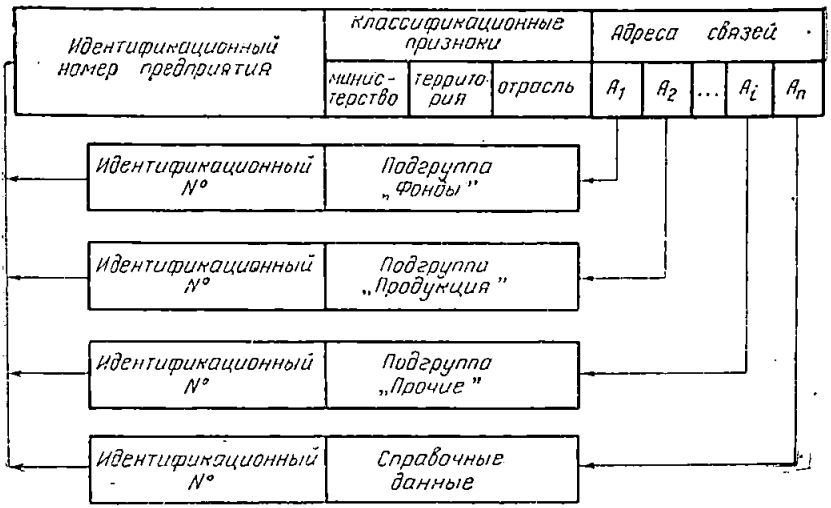



Рис. 1. Взаимосвязь информационных массивов с корневым сегментом регистра

Информационные сегменты должны включать по каждому предприятию следующие блоки: тип сегмента, связку с корневым сегментом ($C_1$); связку со следующим сегментом данного предприятия ($C_2$); связку со следующим однотипным сегментом другого предприятия ($C_3$); блок группы показателей.

Фрагмент взаимосвязи информационной базы и корневого сегмента регистра с использованием адресов-связок представлен на рис. 1.

При такой организации данных структура записей будет зависеть от периода, за который информация накапливается в массиве, и от

числа показателей, включаемых в регистр. Поэтому при изменении периода или числа показателей информационный сегмент группы показателей может иметь различную длину записи. Структуру основного информационного массива РПП в этом случае можно представить в виде совокупности записей матричной формы (рис. 2).

На рисунке введены следующие обозначения:

$P_1, \ldots, P_n$ — идентификационные номера предприятий;

$K_1, \ldots, K_n$ — классификационные признаки;

$g_1, \ldots, g_n$ — группы экономических показателей (фонды, продукция, труд и заработная плата и т. д.);

$a_{11}, \ldots, z_{nm}$ — экономические показатели, формируемые за конкретный период времени (например, за пятилетку).

Данную матрицу можно рассматривать как динамическую информационную модель объекта, в которой каждый столбец характеризует состояние объекта на определенный момент времени, а каждая строка — динамику того или иного показателя. Следовательно, показатели по каждому объекту будут располагаться по годам, образуя таким образом динамический ряд.

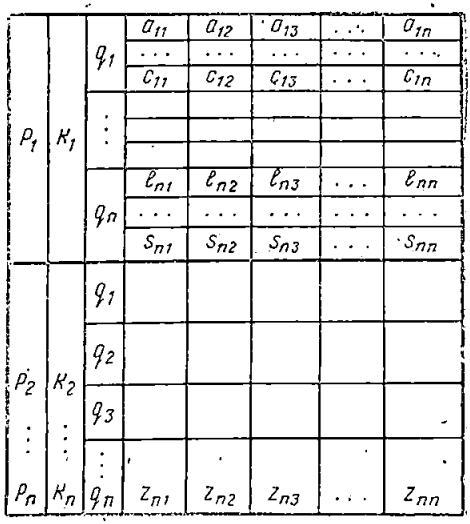


Рис. 2. Структура основного информационного массива РПП

Такая структура регистра позволяет рационально осуществлять поиск данных при наличии соответствующей информации большого количества объектов. Так, с помощью адресов-связок между однотипными информационными сегментами отыскиваются данные, необходимые для решения поставленной задачи, и записываются в рабочий массив со связками с корневым сегментом. После просмотра всех однотипных информационных сегментов, содержащих данные за определенный период времени, на МД формируется рабочий массив. Затем по идентификационным номерам и соответствующим признакам классификации (коды министерства, территории, отрасли) информация сортируется и в требуемом разрезе выдается на печать.

Наиболее перспективным методом формализации описания служит матричное представление взаимосвязей задач и показателей или их совокупностей. Данный метод заключается в том, что для комплекса статистических задач составляется матрица $M_{ij}$, каждая

строка которой соответствует отдельной задаче, а столбец — тематическому массиву (показателю):

| Коды задач | Идентификаторы показателей | | | | | | | | | | | | | Классификационные признаки | | | Идентификационный номер объекта |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | | | | B | | | C | | | | | j | министерство | территория | отрасль | |
| | $A_1$ | $A_2$ | . . . | $A_k$ | $B_1$ | . . . | $B_n$ | $C_1$ | . . . | $C_m$ | | | | | | | |
| 001 | 1 | 0 | | 1 | 1 | | 1 | 0 | | 1 | | | 0 | | | | |
| 002 | 1 | 1 | | 0 | 0 | | 1 | 1 | | 0 | | | 1 | | | | |
| 003 | 0 | 1 | | 1 | 0 | | 0 | 1 | | 0 | | | 1 | | | | |
| 004 | 1 | 1 | | 1 | 0 | | 1 | 1 | | 1 | | | 1 | | | | |
| 005 | 0 | 1 | | 0 | 1 | | 0 | 1 | | 1 | | | 0 | | | | |

Единицы в прямоугольниках указывают на применяемость массивов в задачах.

Матрица $M_{ij}$ составляется заранее на основе анализа соответствующих задач и исходных данных, записывается на МД и используется при формировании рабочих массивов — исходных данных для решения отдельных статистических задач или целого комплекса. Создание рабочих массивов необходимо в связи с тем, что непосредственное использование основных информационных массивов в качестве исходных при решении задач не обеспечивает гибкости системы обработки информации при ее развитии. Использование именно рабочих массивов, формируемых программным путем из основных массивов регистра, позволит обеспечить разработку функциональных задач статистики до того, как будет создан единый банк данных, и с развитием системы осуществить реорганизацию фонда задач и, кроме того, использовать при решении конкретной задачи в качестве исходной информации массивы с минимально необходимыми объемами информации.

С целью формирования рабочих программ обработки запросов организуется матрица задач $B=\|B_{ij}\|$, которая представляет собой набор программных модулей, используемых при решении статистических задач. Структура матрицы задач аналогична структуре матрицы массивов, только каждая строка матрицы $(B_i)$ содержит перечень идентификаторов модулей, столбец $(B_j)$ — перечень статистических задач.

На основании этих матриц определяется, какие модули участвуют в решении каждой конкретной задачи, устанавливается также последовательность решения и взаимосвязь задач. Причем последовательность и теснота взаимосвязи определяются степенью детализации описания данных. При этом установлено следующее: чем де-

тальнее описание, чем больше с его помощью зафиксировано связей и отношений между показателями, тем большее количество ответов можно получить на запросы потребителей. В то же время излишняя детальность описания данных увеличивает трудоемкость выполнения соответствующих процедур. Поэтому в целях ликвидации данного противоречия большое внимание должно уделяться вопросам использования общесоюзных систем обозначений; разработке специального информационного языка, формальный аппарат которого позволял бы рационально описывать данные и их отношения, методы доступа к данным, способы защиты и корректировки данных, обеспечивать удобство доступа пользователей к данным, хранящимся в регистре, особенно в части возможности обращения к РПП с запросом, сформулированным на формализованном естественном языке.

Применение общесоюзных классификаторов и систем обозначений для описания основных характеристик объектов учета позволяет их однозначно описывать не только в рамках АСГС, но, что особенно важно, при межсистемном обмене информацией с другими автоматизированными системами. Требование удобства обращения к РПП во многих случаях оказывается определяющим, поскольку пользователи, как правило, осуществляют связь с регистром через информационный запрос, оформленный на специальном бланке. Следовательно, для пользователя чем проще правила формирования и заполнения бланка запроса, тем удобнее форма общения.

Информационный запрос — это языковое выражение информационной потребности в данных, хранящихся в регистре. Основными элементами любого запроса являются точная формулировка темы запроса, вид выходной информации (таблица, форма, табуляграмма и т. д.), средства доставки и представления данных, степень срочности ответа. Тема запроса включает сведения о периоде, за который следует вести обработку; о показателях, участвующих в решении задачи; о типе объектов учета (предприятие, министерство, отрасль, территория).

Запрос также может включать дополнительные сведения, в частности, принадлежность объекта учета к конкретным министерствам, отраслям, территориям, интервалы значений экономических показателей за какой-либо год при решении группировочных задач. В целях рационального общения потребителя с регистром и экономии памяти машины в запросе указываются только первичные показатели, на основе которых по соответствующим формулам, заложенным в машине, рассчитываются производные показатели, значительно расширяющие информационные возможности регистра.

При матричном представлении и описании данных поступивший запрос анализируется одной из программ диспетчера машины. Основные функции диспетчера при анализе и преобразовании запроса сводятся к принятию решения о том, какие программы и в каком режиме и последовательности необходимо выполнять для того, чтобы данный запрос был удовлетворен. В частности, по данным матрицы массивов $M=\|M_{ij}\|$ проводится анализ показателей $(M_i)$, необ-

ходимых для организации рабочих массивов, а на основе матрицы задач $B=\|B_{ij}\|$ определяется набор модулей $(B_i)$ с целью формирования рабочей программы, реализующей ответ на данный запрос. Другими словами, в работе диспетчера при анализе и преобразовании запроса различают две основные функции: анализ запроса с целью определения режима работы системы и генерирования требуемых последовательностей программ и приведение запроса к виду, необходимому для работы.

Организация и ведение регистра по описанной выше структуре представления данных повышают эффективность при выполнении различных структурных и содержательных преобразований над массивами данных. Основное же преимущество регистровой формы хранения и обработки данных состоит в интегрированном формировании и использовании системы показателей, позволяющих решать комплекс взаимосвязанных экономико-статистических задач. Следовательно, такая форма наблюдения будет способствовать совершенствованию системы показателей, так как регистр позволяет осуществлять объединение показателей целого комплекса статистических отчетов в единую совокупность статистических показателей, всесторонне характеризующих производственно-хозяйственную и финансовую деятельность предприятий.

# 2 ПРОЕКТИРОВАНИЕ И МОДЕЛИРОВАНИЕ СИСТЕМ ОБРАБОТКИ ДАННЫХ

*А. И. МИШЕНИН*

## ПАРАМЕТРЫ, ВЛИЯЮЩИЕ НА ВЫБОР СТРУКТУРЫ ДАННЫХ

Современные СУБД дают пользователям широкие возможности для представления и обработки данных в памяти ЭВМ (структуры данных). Тип структуры данных служит основным фактором для организации поиска и корректировки — наиболее массовых операций в базе данных [2, 3, 5]. Выбор структуры данных является ответственным решением при проектировании системы, от которого существенно зависит эффективность ее функционирования.

В известных работах эффективность структуры данных оценивается либо только при поиске, либо только при корректировке. Между тем необходим совместный анализ этих операций, поскольку лучшая структура данных для поиска не совпадает с лучшей структурой данных для корректировки.

### Постановка задачи

Эффективность процесса поиска и корректировки в различных структурах данных будем оценивать по среднему времени реализации соответствующих процедур на некотором интервале времени. Границами интервала служат два смежных во времени запроса на последовательную обработку данных, когда записи извлекаются из структуры в логической последовательности [1]. За это время выполняется $n$ поисковых и $m$ корректирующих обращений к структуре данных. Будем считать, что они приходят неодновременно и их взаимный порядок безразличен. Время последовательной обработки в критерий не включается (оно сильно зависит от конкретной задачи). Под структурой данных понимается только способ взаимосвязи записей (групп).

Объем памяти для хранения данных в качестве критерия не рассматривается. Общий критерий $K=TV$ ($T$ — время обработки, $V$ — объем памяти) делает изменения $T$ и $V$ равноценными, однако со-

кращение времени обработки данных важнее уменьшения объема памяти при представлении данных.

Для сравнения выбраны упорядоченные структуры данных — последовательная, строчная и древовидная (бинарная), функциональные возможности которых практически одинаковы. Элементами этих структур считаются записи с одним ключевым признаком, по которому выполняется поиск.

Процесс корректировки считается стационарным [2], т. е. корректировка приводит к незначительным отклонениям от среднего числа записей в стабилизировавшихся структурах данных и не может систематически уменьшать или увеличивать их число.

Кроме определенных выше $n$ и $m$, параметрами, от которых зависит критерий эффективности, являются:

$M$ — количество записей в структуре данных;

$L$ — длина одной записи (в словах памяти ЕС ЭВМ);

$m_в$ — количество вновь включенных записей за стандартный период времени.

В стабилизировавшейся структуре данных можно допустить $m_и = m_в$, где $m_и$ — число исключаемых записей. Количество записей с измененными полями $m_з$ за рассматриваемый период времени получается из соотношения $m_в + m_и + m_з = m$, откуда

$$m_з = m - 2m_в \geqslant 0. \qquad (1)$$

Эффективность структур данных анализируется в следующей последовательности: сначала определяется наилучший метод доступа к каждой структуре, а затем эти методы сравниваются между собой.

## Выбор метода доступа для последовательной структуры данных

В рамках поставленной задачи метод доступа указывает на алгоритм поиска, алгоритм корректировки и уточняет наиболее удобное для этих алгоритмов представление структуры данных в оперативной памяти.

При дихотомическом поиске в последовательной структуре данных используется минимально возможное число сравнений [6, 7]. Следовательно, чтобы другие алгоритмы поиска работали еще быстрее, они должны, кроме сравнений, выполнять и другие операции с ключами, например вычисление адресов. Использование известных методов рандомизации усложняет и замедляет последовательную обработку данных и поэтому они не рассматриваются как допустимые методы доступа.

Некоторого ускорения поиска по сравнению с работой алгоритма дихотомии можно добиться при построении для основного массива специального массива $A$ — индексов $A = \{a_1, a_2, \ldots, a_i, \ldots, a_n\}$ [4]. В индексе $a_i$ адресуется запись, ключ которой равен или непосредственно больше значения $p_1 + z(i-1)$, где $p_1$ — ключ первой записи массива, $z$ — константа. Наличие ключа в $A$-индексе необязательно.

Пример $A$-индексов для массива и строки ($z=10$) показан на рис. 1. Поиск ведется в две стадии. Сначала по значению искомого ключа $q$ определяется

$$i = E\left(\frac{q-p_1}{z}\right)+1, \qquad (2)$$

где $E(x)$ — целая часть $x$.

Затем записи, адресованные в индексах $a_i$ и $a_{i+1}$, принимаются за границы интервала, в котором поиск заканчивается методом дихотомии. Время поиска в массиве, если значения ключей распределены равномерно, составляет в среднем

$$T_{\text{ср}} = T_1 + \tau_1 \log_2 \frac{M}{w} < \tau_1 \log_2 M, \qquad (3)$$

где $T_1$ — время вычислений по формуле (2) и обращения к двум $A$-индексам;

$\tau_1$ — время одного цикла в методе дихотомии;

$w$ — число $A$-индексов.

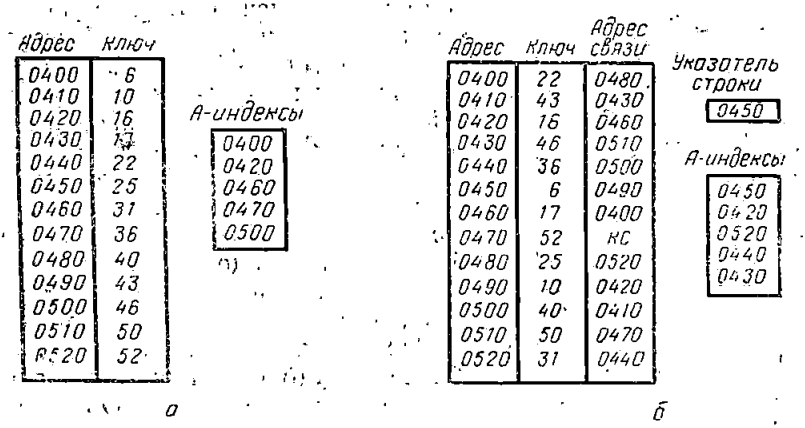



Рис. 1. Пример А-индексов для массива (*а*) и строки (*б*)

Справа в (3) стоит время поиска методом дихотомии. Оно больше $T_{\text{ср}}$ и разница увеличивается с ростом $w$.

Варианты организации корректировки в последовательной структуре данных показаны на рис. 2. Любой путь от входной вершины к выходной определяет некоторый метод корректировки. Число практически важных методов значительно меньше, чем следует из рис. 2. Их оценка выполнена, например, в работе [2].

Для детального сравнения мы выбрали три метода доступа: накопительный (алгоритм которого определяется характеристиками 2, 3, 4, 8, 9, 11, 14, 16), $K$-метод (с характеристиками 2, 3, 4, 6) и $A$-метод (с характеристиками 2, 3А, 5, 6). Их отличительной чер-

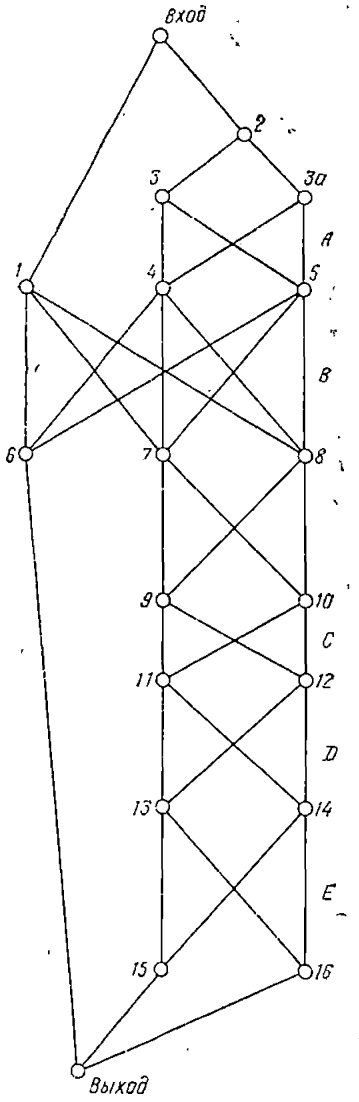



Рис. 2. Варианты организации корректировки в последовательной структуре данных:
А — разделение на части основного массива: 1 — неразделенный; 2 — разделенный; 3 — резерв памяти в каждом блоке; 3а — есть дополнительный резерв, общий для всего массива; 4 — блоки выделяются произвольно; 5 — блоки выделяются согласно А-индексам; В — задержка внесения изменений (наличие массива изменений): 6 — нет задержки; 7 — задержка всех записей (общая); 8 — задержка только по включаемым записям; С — накопление изменений: 9 — изменения организованы последовательно; 10 — изменения связаны в цепь; 11 — изменения упорядочены; 12 — изменения не упорядочены; Д — частота объединения: 13 — по требованию на последовательную обработку; 14 — когда заполнен массив изменений; Е — метод объединения: 15 — с использованием резерва памяти; 16 — без использования резерва памяти

той является использование индексов, обеспечивающих ускоренный поиск.

**Накопительный метод.** Основной массив разделяется на несколько частей необязательно одинаковой длины. Каждая часть сопровождается отдельным массивом изменений, который располагается вслед за нею. Таким образом, основной массив состоит из ряда блоков, а в каждом блоке выделяется основной сегмент и сегмент изменений. Все блоки снабжены индексами.

При поиске определяется номер блока и методом дихотомии просматриваются оба сегмента. Общее время поиска составляет в среднем

$$T_{\text{п}} = \tau_1 n \left(\log_2 k + \log_2 \frac{M}{k} - 1 + \log_2 s - 1\right) = \tau_1 n (\log_2 M + \log_2 s - 2), \qquad (4)$$

где $k$ — число блоков;

$s$ — среднее число записей в сегменте изменений.

Включение новой записи в сегмент изменений означает поиск ее места, сдвиг части записей и пересылку новой записи в освободившийся участок. Это потребует времени

$$T_{\text{в}} = \tau_1 m_{\text{в}} (\log_2 k + \log_2 s - 1) + \frac{L t_1 m_{\text{в}} s}{4}, \qquad (5)$$

где $t_1$ — время пересылки одного слова.

При исключении записи ее ключ принимает максимальное по

формату значение, содержащее цифру $F$ в каждом разряде. Позиции записей, исключенных из основного сегмента, связываются в цепь, причем адреса связи в ней образуют возрастающую последовательность. Время исключения записи из основного сегмента больше, чем из сегмента изменений, поэтому будем учитывать только работу с основным сегментом:

$$T_{\text{и}} = m_{\text{и}} \left[\tau_1 (\log_2 M - 1) + \frac{\tau_2 m_{\text{и}}}{2k^2} + 3t_2 + lt_1\right], \qquad (6)$$

где $\tau_2$ — время одного цикла поиска методом перебора;
$t_2$ — время пересылки одного адреса;
$l$ — длина ключа в словах.

Второе и третье слагаемые описывают поиск и включение освободившейся позиции памяти в цепь. Время замены записей составляет

$$T_{\text{з}} = m_{\text{з}} \left[\tau_1 (\log_2 M - 1) + Lt_1\right]. \qquad (7)$$

Выдача записей на последовательную обработку происходит по схеме, аналогичной алгоритму слияния, и требует времени

$$T_{\text{пос}} = \bar{t}\left(M + \frac{ks}{2}\right), \qquad (8)$$

где $\bar{t}$ — время одного цикла.

При объединении двух сегментов сегмент изменений подставляется в цепочку записей, исключенных из основного сегмента, за время $sLt_1$. Полученный блок сортируется методом вставки [2]. За счет высокой упорядоченности результата объединения (оба исходных сегмента были отсортированы) число циклов сортировки приблизительно равно $M/k$. Число последовательных обработок за период, когда объединяются сегменты во всех $k$ блоках, составляет $ks/m_{\text{в}}$. Поэтому в критерий эффективности время объединения сегментов должно войти в виде

$$T_{\text{об}} = \frac{m_{\text{в}}}{ks} k\left(sLt_1 + \frac{Mt_{\text{в}}}{k}\right) = m_{\text{в}}\left(Lt_1 + \frac{Mt_{\text{в}}}{ks}\right), \qquad (9)$$

где $t_{\text{в}}$ — время одного цикла метода вставки.

Из-за неравномерности включения и исключения записей блок переполняется, когда $m'_{\text{в}} - m'_{\text{и}} > s$ ($m'_{\text{в}}$, $m'_{\text{и}}$ — число включений и исключений от момента создания блока до текущего момента времени). В этом случае создается новый блок и в него помещается половина записей из переполненного блока. Время создания нового блока равно:

$$T_{\text{ф}} = Lt_1\left(\frac{2M}{k} + 2s + \frac{M + ks}{2}\right). \qquad (10)$$

Вероятность переполнения вычисляется в соответствии с моделью, описанной в [8]. Обозначим через $x$ среднее число случаев, когда наступает равенство $m'_{\text{в}} = m'_{\text{и}}$ за период времени, в котором произошло $m'$ включений и исключений записей в блоке. Согласно [8] при $m' = 100$ $x = 6{,}7$. Доказано, что медиана величины $x$ при росте

$m'$ в $d$ раз растет в $\sqrt{d}$ раз. Период, с которым появляется равенство $m'_{в}=m'_{и}$, составляет в среднем $m'/x$, откуда максимальная величина разности

$$m_{в}' - m_{и}' = s = \frac{m'}{2x}. \qquad (11)$$

За время, для которого рассчитывается критерий эффективности, $m'$ достигнет величины $2m_{в}/k$. Чтобы вероятность переполнения не превышала 0,5, надо определить $x$ согласно приведенным выше утверждениям, после чего

$$s = 0{,}76\sqrt{m'} = 1{,}07\sqrt{\frac{m_{в}}{k}}. \qquad (12)$$

Отсюда $1{,}14m_{в}/ks^2=1$. Вероятность переполнения одного блока

$$\beta = \frac{0{,}5 \cdot 1{,}14m_{в}}{ks^2} = \frac{0{,}57m_{в}}{ks^2} \qquad (13)$$

составляет 0,5, если $s$ соответствует формуле (12), и меньше 0,5, если $s$ завышено, причем вывод основан на замене отраженного нормального распределения для $x$ треугольным.

Критерий эффективности накопительного метода доступа имеет вид

$$Э_{п} = T_{п} + T_{в} + T_{и} + T_{з} + T_{пос} + T_{об} + \beta kT_{ф}. \qquad (14)$$

**К-метод.** Все блоки массива имеют длину $M/k+s$ и вначале содержат только по $M/k$ записей. Блоки снабжены индексами. При корректировке сдвигается в среднем половина записей блока. Алгоритм обработки переполнения, время его реализации и оценка вероятности переполнения такие же, как в накопительном методе доступа.

Несколько завышенное время поиска составляет

$$T_{п} = \tau_1 n\,[\log_2(M + ks) - 1]. \qquad (15)$$

Общее время корректировки не превышает величины

$$T_{k} = \tau_1 m\,[\log_2(M + ks) - 1] + \frac{M}{2k}(m_{в} + m_{и})\,Lt_1 + m_{з}Lt_1. \qquad (16)$$

Перед последовательной обработкой основного массива не требуется никаких вспомогательных операций, поэтому критерий эффективности для К-метода равен:

$$Э_{k} = \tau_1 (n + m)\,[\log_2(M + ks) - 1] + \frac{Mm_{в}Lt_1}{k} + m_{з}Lt_1 + \beta kT_{ф}. \qquad (17)$$

К-метод доступа лучше обработки массива, не разделенного на блоки, по алгоритмам, приведенным в [2], если $m_{в}/k<1{,}66$. Когда $m_{в}/k>5$, требуется увеличить $k$, в промежуточных случаях ответ зависит от значения $L$.

**А-метод.** Блоки записей в массиве выделяются в соответствии с интервалами, которые задаются $A$-индексами. Резервная зона рассчитана на $ks$ записей. Данные, логически принадлежащие разным

блокам, образуют в ней несколько цепочек. Поскольку длина цепочки, как правило, невелика, будем считать, что время поиска в блоке всегда больше и составляет

$$T_{\text{п}} = \tau_1 n\,[\log_2(M + ks) - \log_2 w - 1] + T_1 n. \qquad (18)$$

По аналогии с (17) критерий эффективности А-метода равен:

$$\Э_{\text{А}} = \tau_1 (n + m)\,[\log_2(M + ks) - \log_2 w - 1] + \\ + T_1 (n + m) + \frac{M m_{\text{в}} L t_1}{w} + m_{\text{з}} L t_1. \qquad (19)$$

Вероятность переполнения резервной зоны в соответствии с (13) уменьшится в $k$ раз, поэтому возможность переполнения не рассматривается.

Разность $\Э_k - \Э_{\text{А}}$ без учета $\beta k T_{\text{ф}}$ имеет вид

$$(n + m)(\tau_1 \log_2 w - T_1) + m_{\text{в}} L t_1 \left(\frac{M}{k} - \frac{M}{w}\right).$$

Она положительна, когда

$$\log_2 w > \frac{T_1}{\tau_1} \text{ и } w > k. \qquad (20)$$

Разность $\Э_{\text{н}} - \Э_k$ содержит слагаемые, пропорциональные $M$, которые в решающей мере определяют ее знак. Разность положительна, если

$$M\left(\frac{\bar{t}}{m_{\text{з}}} + \frac{t_{\text{в}}}{ks} - \frac{Lt}{k}\right) > 0.$$

Отсюда

$$k > \frac{m_{\text{в}} L t_1}{\bar{t}} - \frac{t_{\text{в}} m_{\text{в}}}{s\bar{t}}. \qquad (21)$$

Проведенное сравнение означает, что последовательные структуры данных надо организовывать А-методом, выбирая $w$ согласно (20—21).

### Оценка эффективности других структур данных

Различных методов доступа к строке известно немного. Для ускорения поиска целесообразно иметь А-индексы. Непрерывный учет свободной памяти при корректировке строки лучше, чем «сборка мусора» [3]. Поэтому критерий эффективности строки имеет вид

$$\Э_{\text{С}} = n\left(T_1 + \frac{\tau_2 M}{2w}\right) + m\left(T_1 + \frac{\tau_2 M}{2w}\right) + \\ + 4t_2 (m_{\text{в}} + m_{\text{н}}) + L t_1 (m_{\text{в}} + m_{\text{з}}). \qquad (22)$$

При выборе модификации бинарного дерева надо учитывать затраты времени и частоту перестройки дерева, поэтому дерево с перестройкой неполных узлов [6] (средняя длина ветви $R = 1{,}20 \log_2 M$) лучше деревьев с меньшим $R$.

Время поиска в рассматриваемом дереве составляет

$$T_{\text{п}} = 1{,}20 n \tau_3 \log_2 M, \qquad (23)$$

где $\tau_3$ — время одного цикла поиска в дереве.

Замена адресов и пересылка записей при корректировке происходит так же, как и в строке. При включении новых записей работа алгоритма перестройки дерева требуется в 29% случаев, а при исключении — несколько реже. Время перестройки можно оценить величиной

$$T_{\text{пер}} = 0{,}29 (m_{\text{в}} + m_{\text{и}}) \tau_{\text{пер}}, \qquad (24)$$

где $\tau_{\text{пер}}$ — время одного запуска алгоритма перестройки.

В итоге критерий эффективности для бинарного дерева определяется выражением

$$\begin{aligned} Э_{\text{Д}} = {} & 1{,}20 (n + m) \tau_3 \log_2 M + 4 t_2 (m_{\text{в}} + m_{\text{и}}) + \\ & + L t_1 (m_{\text{в}} + m_{\text{э}}) + 0{,}29 (m_{\text{в}} + m_{\text{и}}) \tau_{\text{пер}}. \end{aligned} \qquad (25)$$

Сравнение последовательной и древовидной структур данных основано на решении неравенства $Э_{\text{А}} - Э_D > 0$, которое после подстановки (19) и (25) дает

$$\begin{aligned} \frac{m_{\text{в}}}{n+m} L t_1 \left( \frac{M}{w} - \frac{0{,}58 \tau_{\text{пер}} + L t_1}{L t_1} \right) > {} & 1{,}20 \tau_3 \log_2 M - \\ & - \tau_1 \log_2 (M + ks) + \tau_1 \log_2 w + \tau_1 - T_1. \end{aligned}$$

При обычно существующих соотношениях между $\tau_1$, $\tau_3$ и $T_1$ правая часть неравенства практически составляет $\tau_1 \log_2 w$. В левой части второе слагаемое составляет незначительную долю от первого и поэтому может быть опущено. В итоге древовидная структура данных лучше последовательной при соблюдении неравенства

$$L > \frac{\tau_1 w \log_2 w}{t_1 \alpha_{\text{в}} M}, \qquad (26)$$

где $\alpha_{\text{в}} = m_{\text{в}}/(n+m)$ — интенсивность включения записей.

Неравенство $Э_D > Э_{\text{С}}$ без учета времени перестройки в дереве приводит к

$$w > \frac{\tau_2 M}{2{,}40 \tau_3 \log_2 M - 2 T_1}. \qquad (27)$$

Следствием из $Э_{\text{А}} > Э_{\text{С}}$ является соотношение

$$\tau_1 \log_2 \frac{M + ks}{w} > \frac{\tau_2 M}{2w} - \alpha_{\text{в}} L t_1 \left( \frac{M}{w} - 1 \right).$$

Учитывая $M \gg ks$ и $M \gg w$, можно перейти к неравенству

$$L > \frac{\tau_2}{2 t_1 \alpha_{\text{в}}} - \frac{\tau_1 w}{\alpha_{\text{в}} t_1 M} \log_2 \frac{M}{w}. \qquad (28)$$

Таким образом, на выбор одной из трех структур данных влияют 4 параметра — $M$, $L$, $w$ и $\alpha_{\text{в}}$. Графическая иллюстрация для случая $\alpha_{\text{в}} = \text{const}$, $M = \text{const}$ показана на рис. 3. Плоскость графика раз-

деляется на три области, в которых оптимальна одна из трех структур данных.

Практически важными являются два вывода:

1. Если точка ($M$, $L$, $w$, $\alpha_в$) не находится вблизи границы области (рис. 3), то неравенства (26—28) правильно определяют лучшую структуру данных.

2. Если число А-индексов $w$ удовлетворяет (27—28), то строчная структура данных эффективнее двух других. Следовательно, вопрос о динамической перестройке структуры данных, поставленный в последнее время [7], может иметь такое решение. Данные первоначально организуются в строчную структуру, причем $w$ должно обеспечивать ее эффективность при начальных значениях $M$, $L$, $\alpha_в$. Когда с течением времени значения $M$ и $\alpha_в$ станут «дрейфовать» так, что неравенства (27—28) нарушатся, следует увеличить $w$, не изменяя самой строчной структуры. Быстрее всего это достигается, если $z$ — четное число, тогда оно заменяется на $0{,}5z$ (см. (2)).

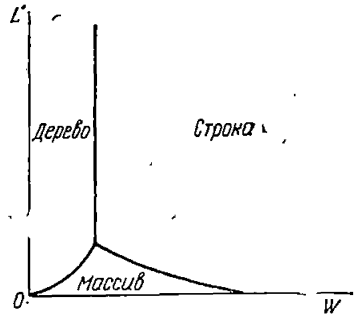



Рис. 3. График границ, определяющих эффективность отдельных структур данных при $\alpha_в$=const и $M$=const

Соотношение между критериями эффективности для структур данных, естественно, будет меняться с появлением новых методов поиска и корректировки, а также новых приемов программирования для известных алгоритмов.

## ЛИТЕРАТУРА


1. Введение в запоминающие устройства прямого доступа и методы организации данных. Пер. с англ. М., Статистика, 1974.

2. Глушков В. М. и др. Обработка информационных массивов в автоматизированных системах управления. Киев, Наукова думка, 1970.

3. Кнут Д. Искусство программирования для ЭВМ. Т. 1. М., Мир, 1976

4. Королев М. А., Клешко Г. Н., Мишенин А. И. Информационные системы и структуры данных. М., Статистика, 1977.

5. Мидоу Ч. Анализ информационных систем. Пер. с англ. М., Прогресс, 1977.

6. Паперное А. А., Подымов В. Я. Методы упорядочения информации в цифровых системах. М., Наука, 1973.

7. Первин Ю. А., Шевякова Т. К. Динамические информационные системы на предприятии. М., Статистика, 1975.

8. Феллер В. Введение в теорию вероятностей и ее приложения. Т. 1. М., Мир, 1964.

*В. Н. ГАРУСОВ, В. Г. ГОЛУБКОВ,*
*В. А. ЛОРМАН*

# ППП ДЛЯ РЕШЕНИЯ В АСУП ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ИТЕРАТИВНЫМИ МЕТОДАМИ С ПРИМЕНЕНИЕМ ДИАЛОГОВОГО АППАРАТА

## Исходные предпосылки

АСУП определяют как человеко-машинные системы, в которые органически включаются интегрированные системы обработки данных с целью автоматизации информационных процессов в управлении, применения математических моделей и методов решения экстремальных задач планирования и управления [1].

В технико-экономическом планировании на промышленном предприятии вопросы применения методов математического программирования находятся пока в стадии исследования, и поэтому на данном этапе следует нацеливаться на модели линейного программирования, как наиболее изученные и оправдавшие себя на практике.

Важнейшим вопросом планирования на промышленном предприятии является определение производственной программы. Модели по определению оптимальной годовой производственной программы предприятия в рамках линейного программирования в настоящее время достаточно разработаны [2, 3]. С другой стороны, в настоящее время появился некоторый разрыв между интенсивностью развития отдельных областей математического программирования и практическим применением результатов этого развития [4]. Поэтому задача состоит не в том, чтобы создавать новые модели и предоставлять соответствующие методы, а в комплексном и систематическом использовании уже имеющихся экономико-математических моделей.

Однако при практическом применении экономико-математических моделей в АСУП встречаются трудности, связанные с отсутствием действительно «системных» программных средств, которые бы являлись технологичными для своей предметной области и позволяли довести модель до такого состояния, в котором ее можно использовать в практике принятия экономических решений.

Для ряда других предметных областей такие системы уже созданы [5, 6, 7]. Характерным является то, что в этих системах имеются целые комплексы средств, предназначенных для хранения, подготовки и отображения данных, а наряду с программами оптимизации имеются средства, позволяющие вести диалог пользователя с системой.

Остановимся на тех особенностях применения методов планирования в АСУП, которые определяют главные концепции в разработке соответствующих программных средств.

*Во-первых,* при планировании на промышленном предприятии необходимо обеспечить:

непрерывность планирования;

динамичность, при которой можно корректировать план без нарушения его целостности и взаимоувязанности;

многовариантность, позволяющую оптимизировать планы по критериям, которые можно легко и быстро менять;

возможность учета субъективного фактора, например мнения и опыта плановиков предприятия, в связи с некоторой условностью модели, вызванной малой степенью полноты охвата экономического объекта (из-за ограниченной размерности решаемых задач и трудности подготовки данных для модели), наличием неформализуемых факторов, относительной неточностью исходных данных, заложенных в модель;

возможность творческого поиска принципиально новых неформальных путей улучшения плана за счет изменения границы допустимой области (обычно в районе уже найденной точки оптимума) [8].

Кроме того, необходимо учесть, что оптимальный план достигается на границах области допустимых значений переменных, поэтому он не может быть принят в качестве реальной производственной программы предприятия (так как даже при незначительном отклонении дефицитных ресурсов от значений, принятых для них в модели, выполнение полученного оптимального плана становится невозможным). Поэтому более естественно применять экономико-математические модели в планировании не просто для получения оптимального плана, а для различных модельных исследований. Исследуя на модели различные условия, ситуации, способы организации производства, можно увидеть упущения в их разработке, оценить лучшие варианты производства той или иной продукции, загрузки оборудования, использования ресурсов и т. п. Набор вариантов возможных производственных программ предприятия, полученный в результате решения задачи с разными критериями оптимальности, с учетом различных ограничений, а также с различными изменениями в параметрах модели, позволит руководству предприятия выбрать правильную экономическую политику при составлении проекта плана, установить задание, руководствуясь лучшим из вариантов, который в наибольшей степени отвечает складывающейся на момент принятия решения ситуации.

Такой постановке вопроса более всего отвечает диалоговая система, позволяющая быстро и удобно менять условия задачи.

Другой вывод заключается в том, что система должна хранить во внешней памяти дерево просмотренных вариантов, в котором содержатся изменения, отличающие друг от друга матрицы условий для смежных вариантов. Система должна позволять пользователю получить информацию о любом варианте дерева и выбрать любой вариант для дальнейшего изменения и просчета [7].

Диалоговая система должна позволять решать задачи в их «естественной» постановке (в терминах оптимального планирования), что дает возможность пользователю, не обладающему специальны-

ми навыками, эффективно использовать достаточно сложную систему.

Для обеспечения диалога на содержательном уровне система должна иметь проблемную ориентацию. Это может быть достигнуто при наличии в ней банка моделей (БМ) и тезауруса, в котором отражены все понятия для моделей, входящих в систему.

*Второй особенностью* применения экономико-математических методов в АСУП являются сложность и трудоемкость информационного обеспечения экономико-математических моделей, которая заключается в следующем:

необходимости агрегирования различных ингредиентов экономико-математической модели, расчета коэффициентов целевых функций и системы ограничений;

в том, что практическая ценность результатов, полученных с помощью экономико-математической модели, очень критична к «стыковке» информации в матрице условий задачи (отсутствие строк, столбцов или элементов в отдельных подматрицах, имеющих строго определенный экономический смысл).

Первая задача аналогична задачам, возникающим в АСУП, и связана с обработкой больших объемов экономических данных, расположенных в информационной базе. Поэтому для ее решения может быть применено либо некоторое общесистемное средство (например, генератор программ совместной обработки данных — ГЕСОМ), либо язык высокого уровня (например, PL/1).

Вторая задача специфична для оптимизационных расчетов и ее решение целесообразно предусмотреть в соответствующем пакете прикладных программ (ППП).

Кроме того, необходимо иметь в виду, что информационное обеспечение АСУП должно строиться на принципах:

единой информационной базы;

гибкости информационной базы, которая предусматривает наличие инструмента, позволяющего создавать и поддерживать любое число постоянных или временных массивов;

комплексности задач и рабочих программ, заключающейся в том, что между задачами планирования возникает постоянный обмен информацией.

В этом случае возможна следующая технология.

Пользователь с помощью какого-либо общесистемного средства вычисляет коэффициенты каждой подматрицы общей матрицы условий оптимизационной задачи и заносит их в информационную базу, а дальнейшую обработку данных, связанную с формированием матрицы, должен взять на себя ППП. Введенная выше концепция банка моделей может существенно облегчить задание пользователем информации о структуре матрицы для ее формирования, так как в этом случае достаточно задать принадлежность решаемой задачи к определенному классу моделей.

*Третьей особенностью* применения экономико-математических методов в АСУП является относительная неточность данных, закладываемых в модель. Поэтому применение точных методов (напри-

мер, модифицированного симплекс-метода с мультипликативным представлением обратной матрицы в ППП «Линейное программирование 2» и ППП «Математическое программирование») здесь не оправдано, так как данные, входящие в модель, не так точны, как это предполагается методом. Обычно это считается недостатком данных, хотя иногда следовало бы считать недостатком метода [4].

Более естественным в такой ситуации является применение итеративных (приближенных) методов, позволяющих при заданной точности получить оптимальное решение за относительно небольшое время.

Кроме того, применение итеративных методов более приемлемо в рамках системы, работающей в диалоговом режиме, так как продолжение счета с ранее полученного решения по конечным методам требует определенных временных затрат всякий раз, когда рассматриваемая модель модифицируется, а итеративные алгоритмы могут продолжить работу с любой точки.

Таким образом, ППП для решения задач оптимального планирования в АСУП должен включать:

банк моделей;

диалоговую систему вариантных расчетов на этих моделях;

гибкую систему формирования матриц соответствующих моделей непосредственно из информационной базы АСУП;

итеративные методы оптимизации.

Кроме этих компонентов, в ППП должны входить и такие традиционные составные части, как послеоптимизационный анализ и вывод результатов решения.

Ниже описываются структура и функции отдельных компонентов такого ППП, который в дальнейшем будем называть ППП «СОЛМИ-ЕС».

## Состав ППП и его функционирование

Из сказанного выше следует, что ППП для решения задач оптимального планирования в АСУП должен обеспечивать выполнение следующих функций:

формирование линейных экономико-математических моделей определенного класса непосредственно из информационной базы АСУП;

вариантные расчеты на этих моделях в режиме диалога.

В соответствии с этим в ППП «СОЛМИ-ЕС» входят два комплекса, работающих под управлением ОС ЕС: генератор программ формирования матриц (ГПФМ) и диалоговая система вариантных расчетов (ДИСВАР). Взаимосвязь этих комплексов показана на рис. 1.

Непосредственно формирование матрицы осуществляет специальная программа формирования матриц (ПФМ), которая должна отвечать следующим основным требованиям:

получать информацию непосредственно из информационной базы АСУП;

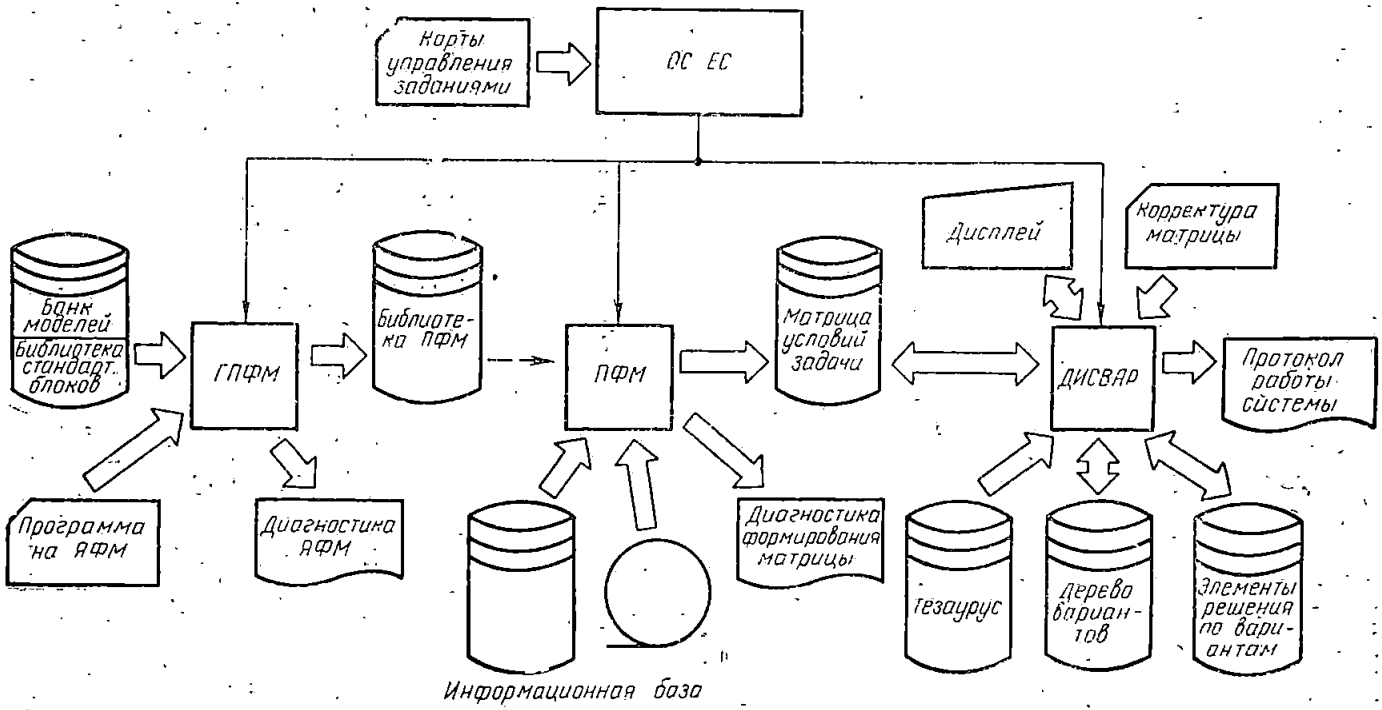



Рис. 1. Взаимосвязь комплексов ППП «СОЛМИ-ЕС»

формировать матрицу условий задачи линейного программирования в рамках любой из линейных экономико-математических моделей, информация о которых хранится в банке моделей пакета.

Из этих требований следует, что ПФМ должна либо настраиваться в процессе выполнения, либо генерироваться по параметрам пользователя. В ППП «СОЛМИ-ЕС» выбран принцип генерации. Это оправдано тем, что ПФМ, предназначенная для работы с конкретной информационной базой, будет использоваться многократно, а постановки задач оптимального планирования в АСУП меняются сравнительно редко.

Для генерации ПФМ служит генератор программ формирования матриц. Параметры пользователя передаются ГПФМ в виде операторов специального языка — языка формирования матриц (ЯФМ). Использование ЯФМ исключает необходимость алгоритмизации процесса формирования и позволяет свести участие специалистов к формальному описанию соответствия между наборами данных информационной базы и подматрицами матрицы условий задачи линейного программирования. Информацию о структуре матрицы условий задачи линейного программирования ГПФМ получает из банка моделей.

Сгенерированная ПФМ формирует матрицу условий и служебную информацию о ней на магнитном диске, а также печатает различные диагностические сообщения о «нестыковках» информации между отдельными подматрицами матрицы условий, если такие «нестыковки» возникли в процессе формирования.

Вторым комплексом пакета является диалоговая система вариантных расчетов, позволяющая по запросам пользователя выполнять следующие действия:

корректировать матрицу условий задачи линейного программирования для ликвидации «нестыковок» информации, если такие «нестыковки» возникли в результате работы ПФМ;

создавать и запоминать во внешней памяти различные варианты матрицы условий задачи, образуемые из исходной матрицы путем удаления, замены или добавления ее отдельных элементов;

решать задачу линейного программирования с любым из ранее созданных вариантов матрицы условий, т. е. «проигрывать» различные варианты основной решаемой задачи;

анализировать устойчивость полученного оптимального решения по отношению к изменению отдельных элементов матрицы условий;

выдавать на печать подробный отчет о результатах решения и послеоптимизационного анализа;

выдавать на экран дисплея любые запрошенные элементы как матрицы условий (коэффициенты целевой функции, коэффициенты и правые части ограничений), так и полученного оптимального решения (значение целевой функции, значения отдельных переменных).

Работа пользователя с комплексом ДИСВАР ведется в режиме диалога через алфавитно-цифровой дисплей; протокол всей работы комплекса (включая протокол диалога) выдается на печать.

Предполагается, что диалог с комплексом ДИСВАР ведет специалист в области планирования, не обладающий особыми знаниями о реализованных в пакете методах оптимизации и о программном обеспечении пакета; вследствие этого диалог ведется в содержательных терминах, хорошо понятных пользователю.

ППП «СОЛМИ-ЕС» функционирует под управлением ОС ЕС следующим образом. Сначала в отдельном пункте задания выполняется генератор программ формирования матриц, который в соответствии с программой, составленной пользователем на языке формирования матриц, генерирует программу формирования матриц и помещает ее в виде загрузочного модуля в библиотечный набор данных, предназначенный для хранения сгенерированных ПФМ.

В другом пункте задания выполняется необходимая ПФМ, в результате работы которой формируется матрица условий задачи линейного программирования, размещаемая на устройстве прямого доступа.

Наконец, в отдельном пункте задания (либо вообще в другом задании) выполняется комплекс ДИСВАР, обеспечивающий решение задачи линейного программирования, матрица условий которой была сформирована с помощью ПФМ. В отличие от выполнения ГПФМ и ПФМ выполнение комплекса ДИСВАР требует присутствия пользователя, ведущего активный диалог с системой.

Рассмотрим теперь более подробно работу каждого из двух комплексов пакета — ГПФМ и ДИСВАР.

### Генератор программ формирования матриц

Выше упоминалось, что информацию о структуре матрицы условий задачи линейного программирования генератор программ формирования матриц получает из банка моделей. В настоящей версии ППП «СОЛМИ-ЕС» в банке моделей предполагается хранить описание пяти экономико-математических моделей. Это общая математическая модель задачи линейного программирования и четыре экономико-математические модели производственного типа:

без учета технологических способов и дополнительных капитальных вложений;

без учета технологических способов, но с учетом дополнительных капитальных вложений;

с учетом технологических способов, но без учета дополнительных капитальных вложений;

с учетом технологических способов и дополнительных капитальных вложений.

В дальнейшем, по мере изучения опыта эксплуатации ППП «СОЛМИ-ЕС», возможно пополнение состава банка моделей.

Описание каждой модели состоит из двух составных частей: описания ограничений модели и описания целевых функций, допускаемых в модели.

Описание структуры моделей основано на том, что ограничения любой экономико-математической модели можно разбить на вертикальные и горизонтальные полосы, состоящие из подматриц, причем каждая полоса и подматрица имеет строго определенный экономический смысл. Каждая подматрица и полоса в банке моделей имеет символическое имя, используемое в операторах ЯФМ, и признак необходимости существования данной полосы или подматрицы в экономико-математической модели.

Описание ограничений и целевых функций из БМ используется в ГПФМ для генерации ПФМ, которая в зависимости от полноты описания модели на ЯФМ будет формировать либо полную конфигурацию матрицы модели заданного класса, либо матрицу одного из допустимых подклассов. Кроме того, описание ограничений и целевых функций модели используется в ГПФМ для определения возможных логических ошибок в программе на ЯФМ.

*Язык формирования матриц*

При описании синтаксиса языка используются металингвистические формулы Бэкуса — Наура.

*Синтаксис*

1. <оператор начала программы на ЯФМ>: :=*НАЧАЛО : : ИМЯ=<имя программы>, МОДЕЛЬ=<имя экономико-математической модели>;
<имя программы>: :=<последовательность алфавитно-цифровых символов>
<имя экономико-математической модели>: :=<имя одной из экономико-математических моделей в банке моделей>
2. <оператор описания структуры записи файла>: :=*ФАЙЛ : : ИМЯ=<имя файла>, <структура записи файла>;
<имя файла>: :=<dd — имя языка управления заданиями ОС ЕС>
<структура записи файла>: :=<описание структуры на языке PL/1>
3. <оператор описания программ пользователя>: :=*ПРОГП : : ИМЯ=<имя программы пользователя>, ЧЗАП=<целое число>, <структура записи файла>
4. <оператор описания подматриц>: :=*ПМАТР: ИМЯ=<имя подматрицы в экономико-математической модели>, [ИМЯП= =<имя подматрицы, данное пользователем>,]<ссылка>, СТРК=<список реквизитов>, СТЛБ=<список реквизитов>, ЭЛЕМ=<имя реквизита>;
<ссылка>: :=ФАЙЛ=<имя файла>/ПРОГП=<имя программы пользователя>
<список реквизитов>: :=<литерал>/<имя реквизита>/
<список реквизитов><знак сцепления><список реквизитов>
<литерал>: :='<последовательность буквенно-цифровых и специальных символов>'
<знак сцепления>: :=!!

5. <оператор списка>: :=*СПИСОК:ИМЯ=<имя полосы в экономико-математической модели>, <ссылка>,
КОД=<список реквизитов>, ТИП=<признак>;
<признак>: :=В/Н
6. <оператор связывания кодов с наименованиями>: :=
=*НАИМЕН:ИМЯ={<имя полосы в экономико-математической модели>,}
<ссылка>, КОД=<список реквизитов>, НАИМ=<имя реквизита>;
7. <оператор задания условий включения в матрицу строк подматриц>: :=*СТРК:ИМЯ=<имя полосы в экономико-математической модели>(<имя подматрицы>{,<имя подматрицы>})
<логическая переменная>{,<логическая переменная>}
{/<логическая переменная>{,<логическая переменная>}};
<логическая переменная>: :=ДА/НЕ/ХХ
8. <оператор задания условий включения в матрицу столбцов подматриц>: :=*СТЛБ:ИМЯ=<имя полосы в экономико-математической модели> (<имя подматрицы>{,<имя подматрицы>})
<логическая переменная>{,<логическая переменная>}
{/<логическая переменная{,<логическая переменная>}};
9. <оператор конца программы на ЯФМ>: :=*КОНЕЦ;

*Семантика*

Особенностью языка формирования матриц является его непроцедурность, т. е. порядок появления операторов ЯФМ никак не влияет на алгоритм формирования матрицы условий (за исключением операторов *НАЧАЛО и *КОНЕЦ, которые определяют начало и конец программы соответственно). В операторе *НАЧАЛО указывается также некоторая дополнительная информация. В частности, в операнде ИМЯ указывается имя, с которым ПФМ будет помещена в библиотеку загрузочных модулей ППП «СОЛМИ-ЕС».

Одним из основных операторов ЯФМ является оператор *ПМАТР. Посредством этого оператора происходит логическое связывание подматрицы экономико-математической модели (операнд ИМЯ) с набором данных информационной базы, из которого формируется эта подматрица (операнд ФАЙЛ или ПРОГП).

В операнде ЭЛЕМ данного оператора указываются реквизиты, образующие коэффициенты подматрицы, в операндах СТРК и СТЛБ указываются реквизиты, образующие имена строк и столбцов соответственно.

Программа формирования матриц может непосредственно обращаться только к наборам данных, организованным в виде линейных структур с однородными записями фиксированной длины. Если же информационная база АСУП представляет базу данных, то обращение к ней осуществляется через программы пользователя. Эти программы должны обеспечивать последовательную выборку записей из базы данных; для этого программа пользователя может, на-

пример, обращаться к системе управления базой данных (СУБД), обслуживающей базу данных. Имя программы пользователя, число (операнд ЧЗАП) и структура передаваемых в ППП «СОЛМИ-ЕС» записей указываются в операторе *ПРОГП.

В ряде случаев пользователю необходимо выбрать из информационной базы для включения в матрицу условий не все изделия, оборудование, материалы и т. д., а только некоторое их подмножество в соответствии с заданным перечнем. Для этого предназначен оператор *СПИСОК, в котором для каждой полосы экономико-математической модели (операнд ИМЯ) указывается ссылка на перечень имен строк (столбцов), подлежащих (ТИП=В) или не подлежащих (ТИП=Н) включению в матрицу.

При выборке имен строк и столбцов для включения их в матрицу наряду с оператором *СПИСОК могут использоваться также операторы *СТРК и *СТЛБ. В этих операторах для каждой полосы экономико-математической модели (операнд ИМЯ) указываются условия включения строк и столбцов в матрицу в зависимости от наличия их в подматрицах этой полосы. Каждое условие представляет собой последовательность логических переменных, число и порядок которых соответствуют перечню имен подматриц полосы, указанному в этих операторах (*СТРК и *СТЛБ).

Поясним сказанное на примере. Пусть задан следующий оператор:

*СТРК : ИМЯ=П2(M,F,S) НЕ,ДА,НЕ/ДА,XX,XX;

Этот оператор означает, что в полосу П2 матрицы условий должны быть включены те строки, которые отвечают любому из двух условий:

строка присутствует (ДА) в подматрице F, но отсутствует (НЕ) в подматрицах M и S;

строка присутствует (ДА) в подматрице M, а наличие или отсутствие ее в подматрицах F и S роли не играет (на это указывают значения логических переменных «XX»).

Во всех остальных случаях строки подматриц M, F и S в матрицу не включаются.

Для ведения содержательного диалога в процессе функционирования комплекса ДИСВАР может понадобиться переход от имен строк и столбцов, в качестве которых обычно выступают коды изделий, групп оборудования и т. д., к наименованиям из классификаторов АСУП. Связь между кодами и наименованиями задается в операторе *НАИМЕН, в котором для полос экономико-математических моделей указывается ссылка на соответствующие классификаторы.

В заключение следует отметить, что не все типы операторов обязаны присутствовать в программе на ЯФМ. Если в программе отсутствуют операторы *СТРК и *СТЛБ для какой-либо полосы, то предполагается, что для всех подматриц, входящих в данную полосу, соответствующие логические переменные имеют значение «ДА». В случае отсутствия других операторов никаких стандартных предположений не делается.

### Функциональная схема ГПФМ

Генератор программ формирования матриц представляет собой отдельный функциональный блок обрабатывающей части ППП «СОЛМИ-ЕС» и состоит из четырех крупных частей, две из которых являются оригинальными разработками (Транслятор с ЯФМ и Планировщик), а две другие относятся к стандартным обрабатывающим программам ОС ЕС (Компилятор PL/1 и Редактор связей). Общая функциональная схема ГПФМ показана на рис. 2.

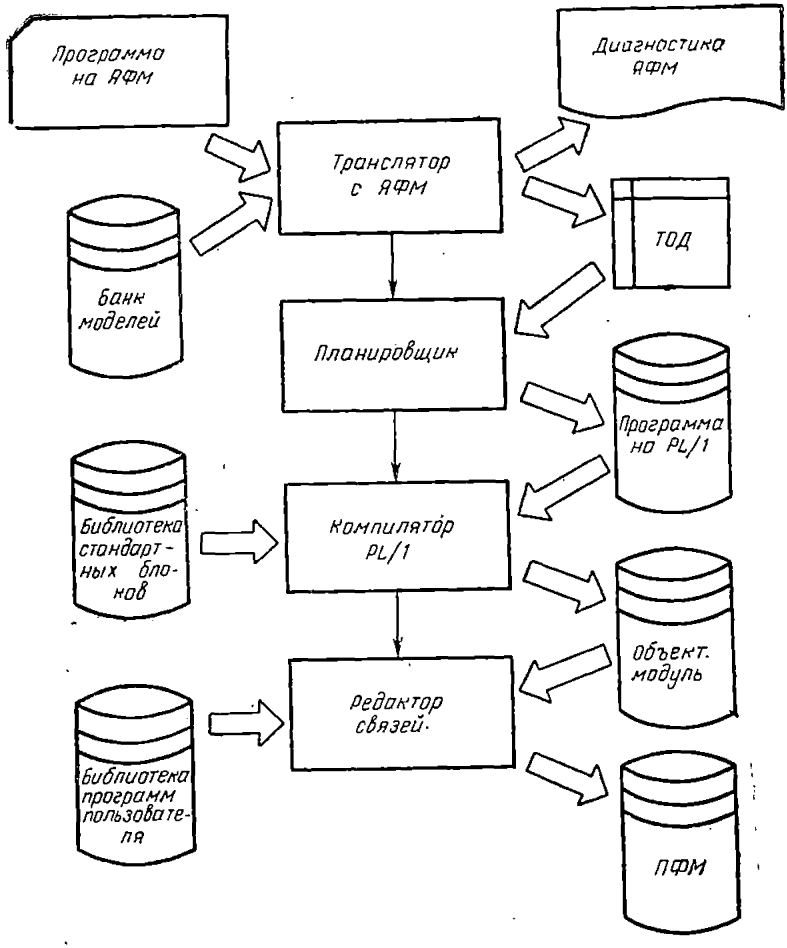



Рис. 2. Функциональная схема ГПФМ

Первым этапом работы генератора программ формирования матриц является синтаксический и семантический анализы программы на ЯФМ. Это осуществляется программой ТРАНСЛЯТОР, которая создает специальные таблицы описания данных (ТОД), являю-

щиеся внутренним отображением заданного пользователем описания задачи.

Следующим этапом работы ГПФМ являются анализ полученных таблиц и построение алгоритма формирования матриц. Этот этап работы реализуется программой ПЛАНИРОВЩИК, которая в зависимости от содержания таблиц формирует те или иные препроцессорные утверждения языка PL/1. Эти препроцессорные утверждения необходимы для вызова стандартных обрабатывающих блоков и настройки их на конкретные параметры пользователя.

Вызываемые обрабатывающие блоки представляют собой функциональные программы на языке PL/1, хранящиеся в виде исходных текстов в специальной библиотеке на устройстве прямого доступа. Настройка стандартных блоков на параметры пользователя достигается применением специальных препроцессорных переменных, которым программа ПЛАНИРОВЩИК присваивает конкретные значения в соответствии с описанием задачи на ЯФМ.

Результатом работы ПЛАНИРОВЩИКА является программа на языке PL/1, содержащая препроцессорные и непрепроцессорные утверждения. Эта программа записывается на устройство прямого доступа и затем обрабатывается компилятором PL/1 (включая этап препроцессорной обработки), в результате чего получается объектный модуль.

Заключительным этапом работы ГПФМ является обработка объектного модуля Редактором связей. При этом происходит подключение программ пользователя, содержащихся в виде объектных модулей в специальной библиотеке. Результатом работы Редактора связей является готовая к выполнению программа формирования матриц. Эта программа является загрузочным модулем и хранится в библиотеке загрузочных модулей ППП «СОЛМИ-ЕС».

### Функциональная схема комплекса ДИСВАР

Комплекс ДИСВАР включает несколько компонентов: Резидент, Монитор и функциональные модули, к которым относятся Корректор, Вариатор, Загрузчик, Оптимизатор, Анализатор, Вывод решения и Лингвистический процессор. Все эти компоненты представляют собой загрузочные модули, вызываемые для выполнения из библиотеки пакета.

Взаимосвязи по управлению и по данным между компонентами комплекса ДИСВАР показаны на рис. 3.

Рассмотрим назначение и функции всех этих компонентов.

*Резидент* является первым вызываемым модулем комплекса ДИСВАР. Он присутствует в оперативной памяти в течение всего времени работы ДИСВАР (отсюда и название этого модуля) и предназначен для вызова Монитора и отдельных функциональных модулей (в соответствии с последовательностью вызова, определенной Монитором, — об этом см. ниже), а также для анализа кодов возврата вызванных модулей и для завершения работы комплекса ДИСВАР (по указанию пользователя или в результате аварийного завершения какого-либо функционального модуля).

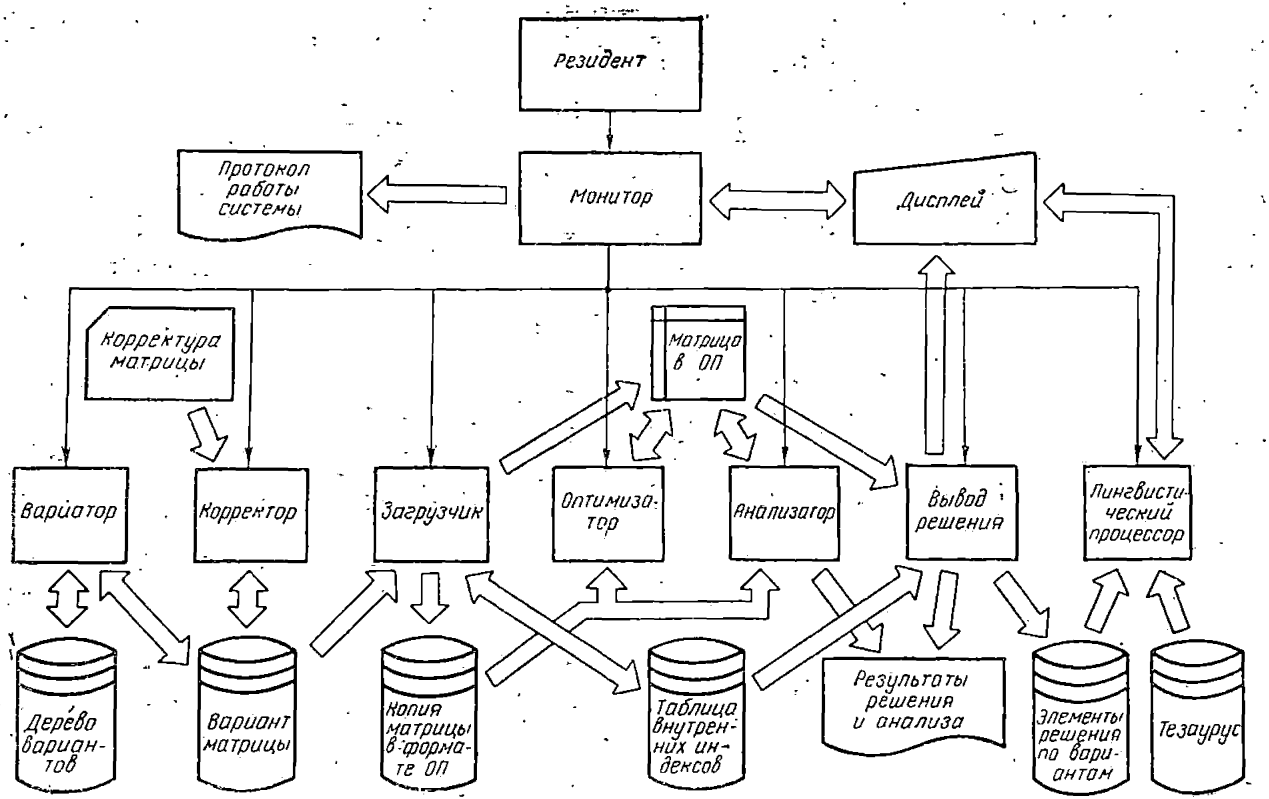



Рис. 3. Функциональная схема комплекса ДИСВАР

*Монитор* является главным управляющим модулем комплекса ДИСВАР. Он обрабатывает поступающие с дисплея директивы пользователя, анализирует их семантику и синтаксис, при наличии ошибок в директиве запрашивает ее повторно; определяет, какие функциональные модули должны быть вызваны для реализации поступившей директивы, строит последовательность вызова этих модулей (например, для реализации директивы «Реши вариант номер 5» необходимо последовательно вызвать Вариатор, Загрузчик, Оптимизатор и Вывод решения); определяет на основании содержания директивы, какие параметры нужно передать вызываемым модулям, сообщает последовательность вызова модулей и передаваемые им параметры Резиденту и передает управление первому модулю из вызываемой последовательности (или единственному модулю, реализующему поступившую директиву). Кроме того, Монитор ведет выдаваемый на печать протокол работы комплекса ДИСВАР.

*Корректор* служит для ликвидации «нестыковок» информации в матрице условий задачи линейного программирования, если такие «нестыковки» возникли в результате работы ПФМ. Ликвидация «нестыковок» осуществляется путем удаления лишних или добавления недостающих элементов в отдельных подматрицах матрицы условий в соответствии с директивами пользователя на перфокартах (при большом числе «нестыковок» обработка колоды перфокарт будет эффективнее, чем обработка отдельных директив с дисплея). Параллельно с изменением матрицы условий на устройстве прямого доступа Корректор соответствующим образом изменяет и служебную информацию о матрице, а также выдает на печать сведения о скорректированной матрице.

*Вариатор* предоставляет пользователю возможность создавать различные варианты матрицы условий задачи линейного программирования путем замены, удаления или добавления отдельных элементов матрицы или путем удаления и добавления любых строк и столбцов матрицы условий. Новый вариант матрицы создается в соответствии с директивами пользователя, вводимыми с дисплея. Вариатор запоминает информацию обо всех создаваемых вариантах матрицы, о логических взаимосвязях между этими вариантами на устройстве прямого доступа, что дает возможность пользователю возвращаться к ранее созданным вариантам и создавать новый вариант на основе любого ранее созданного, т. е. получать дерево вариантов матрицы; по запросу пользователя изображение дерева вариантов может быть выдано на печать или на экран дисплея. С помощью Вариатора любой из ранее созданных вариантов матрицы может быть сделан активным, т. е. доступным для его загрузки в память и решения задачи линейного программирования с данным вариантом матрицы условий. В каждый момент времени имеется единственный активный вариант матрицы.

*Загрузчик* предназначен для переноса (загрузки) активного варианта матрицы с устройства прямого доступа в оперативную память для дальнейшего решения задачи линейного программиро-

вания с данным вариантом матрицы условий с помощью Оптимизатора.

Загрузчик преобразует матрицу условий из того вида, в котором она хранилась на устройстве прямого доступа, к виду, используемому Оптимизатором (при этом, в частности, коды строк и столбцов матрицы заменяются их индексами, т. е. порядковыми номерами, которыми пользуется реализованный в Оптимизаторе метод оптимизации).

На основе анализа служебной информации о матрице условий задачи Загрузчик определяет необходимый размер оперативной памяти для размещения матрицы, после чего пытается получить от ОС ЕС область памяти вычисленного размера с помощью макрокоманды GETMAIN. Если область памяти запрошенного размера есть в наличии, Загрузчик переносит в эту область всю матрицу условий задачи линейного программирования (в виде, пригодном для обработки Оптимизатором). Если же запрос на область памяти вычисленного размера не может быть удовлетворен (т. е. матрица не умещается целиком в оперативной памяти), Загрузчик создает полную копию матрицы на устройстве прямого доступа, но в том виде, который используется Оптимизатором (т. е. в виде отображения оперативной памяти). В этом случае при решении задачи линейного программирования матрица условий вводится с магнитного диска в оперативную память по частям, а Оптимизатор последовательно обрабатывает вводимые части матрицы.

*Оптимизатор* предназначен для нахождения оптимального решения поставленной задачи линейного программирования. Для поиска оптимального решения в нем используются итеративные (приближенные) методы оптимизации, а именно модифицированный метод Петжиковского в сочетании с модифицированным методом Ньютона. Применение этих методов обусловлено, во-первых, возможностью получения решения быстрее, чем точными методами (при этом относительная неточность полученного оптимального решения оправдывается неточностью исходных данных решаемой задачи), а во-вторых, способностью итеративных методов начинать поиск оптимального решения с любой точки, что важно в случае многократного изменения входных данных в процессе «проигрывания» различных вариантов матрицы условий задачи.

Если матрица условий задачи целиком умещается в оперативной памяти, то решение будет получено значительно быстрее, чем когда матрица вводится по частям с магнитного диска. В последнем случае на каждой итерации необходимо последовательно ввести и обработать все части матрицы; именно поэтому копия матрицы хранится на МД в виде отображения оперативной памяти, чтобы избежать больших потерь времени на преобразование матрицы к виду, обрабатываемому Оптимизатором. Хотя использование копии матрицы на МД весьма замедляет получение оптимального решения, оно позволяет даже на вычислительных установках с относительно малым объемом оперативной памяти решать оптимизационные задачи достаточно большой размерности.

По желанию пользователя решение задачи может быть прервано в любой момент с запоминанием текущего состояния, а затем (в более удобное время) продолжено с точки, где оно было прервано.

*Анализатор* служит для анализа чувствительности полученного оптимального решения по отношению к изменению коэффициентов целевой функции или правых частей ограничений. Анализатор определяет, в каких диапазонах можно изменять значения коэффициентов целевой функции и правые части ограничений задачи линейного программирования, чтобы при этом структура оптимального решения оставалась неизменной.

*Вывод решения* обеспечивает выдачу на печать подробной (либо, по желанию пользователя, сокращенной) информации о полученном оптимальном решении. Значения целевой функции и вошедших в оптимальный план переменных выдаются также на экран дисплея. Кроме того, элементы оптимального решения, полученного для каждого варианта матрицы условий задачи, запоминаются на устройстве прямого доступа и впоследствии могут быть использованы Анализатором (для анализа чувствительности решения по заданному варианту матрицы) и Лингвистическим процессором (для выборочной выдачи этих элементов решения пользователю или для расчета некоторых производных величин на основе элементов решения).

*Лингвистический процессор* предназначен для обработки запросов пользователя на выдачу различной информации о решаемой задаче. Требуемая информация выводится на экран дисплея.

Могут быть выданы такие данные для различных вариантов модели, как отдельные элементы матрицы, решения и послеоптимизационного анализа, перечень лимитирующих ресурсов и их двойственных оценок, величина резерва по какой-либо группе ресурсов и т. д.

Кроме того, пользователь может изменять матрицу модели, задавая эти изменения в обобщенном виде. Например, «Увеличить фонды по лимитирующим ресурсам на 3 процента».

Запросы пользователя и выдаваемая ему информация состоят из содержательных терминов, таких, как «прибыль», «объем выпуска», «резерв» и т. д. Для обеспечения содержательного диалога Лингвистический процессор использует тезаурус, включающий все термины, связанные с решением задачи линейного программирования и с применяемыми в пакете экономико-математическими моделями.

Запросы пользователя в отличие от директив, выполняемых другими функциональными модулями, могут иметь относительно произвольный (не фиксированный жестко) вид. Поэтому на Лингвистический процессор возлагается особая функция лингвистического анализа поступающих запросов для выявления их семантики и выполнения соответствующих действий. Этим и объясняется название «Лингвистический процессор».

*Язык диалога с линейными экономико-математическими моделями*

Взаимосвязь между пользователем и диалоговой системой вариантных расчетов (ДИСВАР) осуществляется сообщениями. Сообщением называется логическая единица данных, обрабатываемая ДИСВАР. В зависимости от направленности сообщения разделяются на входные (от пользователя к системе) и выходные (из системы к пользователю).

Входные сообщения могут быть либо директивами, либо запросами, либо данными. С помощью директив пользователь указывает системе, какие нужно выполнить действия (например, изменить матрицу условий, решить задачу с конкретным вариантом матрицы и т. п.). С помощью запросов пользователь может получить различные данные о матрице, дереве вариантов, элементах решения и анализа из информационной базы ППП. В системе различаются форматизованные и неформатизованные запросы. Форматизованные запросы обрабатывает Вариатор.

Например,

ВЫДАЙ СТРОКУ <код строки>
ВЫДАЙ ЭЛЕМЕНТ <код строки> , <код столбца>

Система допускает запросы, на которые не накладывается особых ограничений по формализации и форматизации. Такие запросы обрабатывает Лингвистический процессор.

Все остальные входные сообщения, кроме директив и запросов, рассматриваются как данные. Пользователь вводит данные после разрешения или запроса со стороны системы.

Выходными сообщениями могут быть ответы на запросы, информация об ошибках во входных сообщениях, а также указания для пользователя.

Более подробно остановимся на неформатизованных запросах. Для обработки таких запросов необходима выработка принципов их формализации системой. В основу формального представления запроса может быть положено следующее предположение [9]: любой элементарный смысл может быть задан как сочетание $(x_i r_j x_k)$, где $x_i$, $x_k$ — термины; $r_j$ — семантическое отношение между терминами.

Роль значений предметных переменных играют единицы естественного языка (слова, выражения), обозначающие элементы экономико-математических моделей и понятия, связанные с вариантными расчетами на этих моделях.

Основой для построения языка запросов является содержательное описание понятий в линейных экономико-математических моделях и связей между этими понятиями.

В результате анализа содержательного описания моделей можно выделить классы понятий. Существуют базовые, неопределяемые понятия: например, «модель», «ограничения», «критерий оптимальности» и др. Производное понятие определяется как множество по-

нятий (базовых или производных), находящихся к уже определенному понятию в отношениях из заданного множества.

Рассмотрим пример формализации понятия «прибыль». «Прибыль есть стоимостной критерий оптимальности». В определении этого термина лексема «стоимостной» сочетается грамматически с лексемой «критерий оптимальности», что в семантическом аспекте интерпретируется как «отношение свойства, качества, любого отличительного признака к предмету».

Или, если обозначить

$x_1$ — прибыль;

$x_2$ — стоимостной;

$r_2$ — отношение свойства, качества, любого отличительного признака к предмету;

$x_3$ — критерий оптимальности,

то формально определение понятия «прибыль» можно записать так:

$$x_1 = x_2 r_2 x_3.$$

Выражение $\{x_i r_j x_k\}$ будем называть семантическим кодом, или RX-кодом. Приведем пример формализации фрагмента запроса: «лимитирующие ресурсы, входящие в состав модели». Обозначим через $x_1$ — лексему «лимитирующий»; $x_2$ — лексему «ресурсы»; $x_3$ — лексему «модель»; $r_2$ — отношение одного предмета к другому предмету, в состав которого входит первый предмет; $r_1$ — отношение свойства, качества, любого отличительного признака к предмету. Тогда RX-кодом для фрагмента приведенного запроса будет цепочка $x_1 r_1 x_2 r_2 x_3$. Каждый запрос может быть представлен как совокупность RX-кодов, в частном случае содержащая один код.

Следуя работе [10], такая совокупность приводится не в виде иерархического дерева, а как последовательность перенумерованных кодов. В этой последовательности особая роль принадлежит последнему коду (ему соответствует наибольший номер), содержащему ядро запроса. В зависимости от того, какие элементы семантического кода входят в состав ядра запроса, последний относится к одному из следующих типов:

нейтральный (ядро запроса занимает позицию отношения $r_j$ и его правого компонента $x_k : x_i — —$);

предметный незаполненный (ядро запроса занимает позицию незаполненного компонента $x_k : x_i r_j —$);

предметный заполненный (ядро запроса занимает позицию заполненного компонента $x_k : x_i r_j x_k$).

Работа Лингвистического процессора происходит следующим образом. Все термины и семантические отношения между ними имеют свои номера в тезаурусе. Транслятор Лингвистического процессора, используя тезаурус, проводит индексирование запроса и определяет принадлежность запроса к тому или иному типу (тем самым строится код запроса). По полученному коду запроса Лингвистический процессор вызывает ту или иную исполнительную программу и передает ей соответствующие параметры. Ответ на запрос выводится на экран дисплея.

* * *

Как было показано выше, ППП «СОЛМИ-ЕС» имеет такие важные достоинства, как:

функциональная направленность;

наличие гибкой системы формирования матриц задач линейного программирования из информационной базы пользователя;

возможность выполнения различных вариантных расчетов в режиме диалога, ведущегося в содержательных экономических терминах.

Эти достоинства пакета позволят достаточно широко использовать его в различных АСУП.

ЛИТЕРАТУРА

1. Глушков В. М. Введение в АСУ. Киев, Техника, 1972.
2. Данилин В. И. Экономико-математические модели годового планирования на предприятии. М., Наука, 1975.
3. Герасимов Н. И. Планирование производственной программы машиностроительного предприятия. М., Экономика, 1972.
4. Канторович Л. В., Романовский И. В. Оптимизационные методы в экономике: результаты, трудности, перспективы. — Управляющие системы и машины, 1977, № 2.
5. Вычислительные методы выбора оптимальных проектных решений. Киев, Наукова думка, 1977.
6. Хачатуров В. Р. и др. Система проектирования генеральных схем обустройства нефтяных месторождений на ЭВМ. Системы программного обеспечения решения задач оптимального планирования. IV Всесоюзный симпозиум. Краткие тезисы докладов. Секция 1 (АН СССР ЦЭМИ). М., 1976.
7. Глушков В. М., Олеярш Г. Б. Диалоговая система планирования ДИСПЛАН. — Управляющие системы и машины, 1976, № 4.
8. Глушков В. М. О диалоговом методе решения оптимальных задач. — Кибернетика, 1975, № 4.
9. Информационно-поисковая система БИТ. Под ред. А. А. Стогния. Киев, Наукова думка, 1968.
10. Труб В. М. К проблеме построения полной типологии стратегий поиска в ИПС. Лингвистические вопросы проектирования и информационный анализ автоматизированных информационных систем. Киев, ИК АН УССР, 1976.

*П. П. ВАСИЛЬЕВ, Ю. В. ЕВСЕЕВ*

## ПРОИЗВОДИТЕЛЬНОСТЬ ТРУДА ПРОГРАММИСТОВ

Задача планирования работ программистов является достаточно трудной, во-первых, из-за отсутствия проверенных и обоснованных нормативов и, во-вторых, из-за чрезвычайно большого разнообразия самих работ как по характеру, так и по трудоемкости. В настоящей статье использован опыт создания программного обеспечения автоматизированной системы учета разработки и реализации программ стандартизации. Существенным является тот факт, что задачи управления в области стандартизации весьма специфич-

ны, вследствие чего невозможно заимствование созданных программных комплексов АСУ. В частности, наблюдается большое разнообразие структур и информационного состава входных и выходных документов и динамичность решаемых задач, а значит нестабильность выходных документов.

Необходимое программное обеспечение, наиболее отвечающее условиям эксплуатации, было выполнено в виде двух комплексов программ.

Первый комплекс содержит программы ввода информации, ее контроля и необходимой структурной перестройки, программы построения и корректировки информационных массивов системы, располагаемых на магнитных лентах. Второй комплекс включает библиотеку программных модулей, реализующих определенные, в некотором смысле элементарные операции обработки информации. Связь комплексов программ осуществляется через структуру информационных массивов.

Для определения множества программных модулей вычислительные алгоритмы решения задач системы были представлены в виде последовательностей операторов, характеризуемых определенными действиями над информационными данными. Такое представление алгоритмов решения задач позволило построить конечное множество операторов, реализующих более 120 элементарных операций по обработке информационных массивов, отдельных их записей и полей. Построенное множество операторов было реализовано на ЭВМ «Минск-32» через функциональные программные модули обработки информации.

В течение 260 дней группа в составе 9 программистов различной квалификации составила 44 программы. Объем выполненных каждым программистом работ, затраченное время и значения средней производительности труда программистов приведены в табл. 1. Следует заметить, что производительность определялась при условиях, когда разрабатывались блок-программы для решения задач некоторого определенного класса, и в ряде программ использовались эти блоки.

Таблица 1

| Шифр программиста | Число составленных программ, шт. | Время работы над программами, дни | Объем выполненной работы, команды | Размер программы в среднем, команды | Стаж работы, годы | Производительность, команд/день |
|---|---|---|---|---|---|---|
| А | 14 | 220 | 5430 | 388 | 10 | 24.7 |
| Б | 4 | 250 | 2507 | 627 | 8 | 10.0 |
| В | 1 | 160 | 1249 | 1249 | 7 | 7.82 |
| Г | 9 | 230 | 3497 | 390 | 6 | 15.2 |
| Д | 3 | 220 | 1709 | 567 | 6 | 7.78 |
| Е | 4 | 260 | 1247 | 313 | 4 | 4.81 |
| Ж | 3 | 260 | 2426 | 810 | 3.5 | 9.27 |
| З | 2 | 260 | 1677 | 838 | 1 | 6.47 |
| И | 4 | 190 | 2361 | 590 | 2.5 | 12.4 |

На основании экспериментальных данных получены зависимости производительности труда программистов $V$ от стажа работы составителя программы $S$ и размера программы $R$. При этом не учитывались корреляционные связи квалификации программиста и размера составляемой программы.

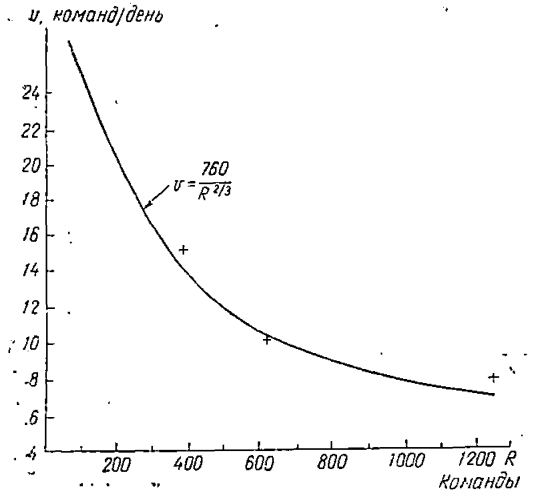



Рис. 1. Зависимость производительности труда программистов от размера составляемой программы

Рассмотрим отдельно влияние каждого исследуемого параметра $S$, $R$ на производительность $V$. С этой целью сгруппируем так полученные экспериментальные данные, чтобы выделить зависимости только от одного параметра при неизменном втором.

Приближенно можно считать, что программисты Б, В, Г и Д имеют одинаковую квалификацию. Тогда полученные экспериментальные данные можно аппроксимировать аналитической зависимостью $V=\frac{760}{R^{2/3}}$ (рис. 1).

Для получения зависимости $V$ от $S$ можно рассмотреть данные по программистам А, Г, Д и Е. Показанная на рис. 1 зависимость $V$ от $R$ может быть интерпретирована следующим образом. Программы по обработке информации, если они небольшого размера, как правило, имеют простую логику. И доводка их до рабочего состояния проводится за 1—3 отладки. Значительные же по размеру программы обычно строятся в виде блоков, отладка которых осуществляется автономно. Вследствие этого в области программ малых размеров производительность резко падает при увеличении размера программы, а начиная с некоторых размеров про-

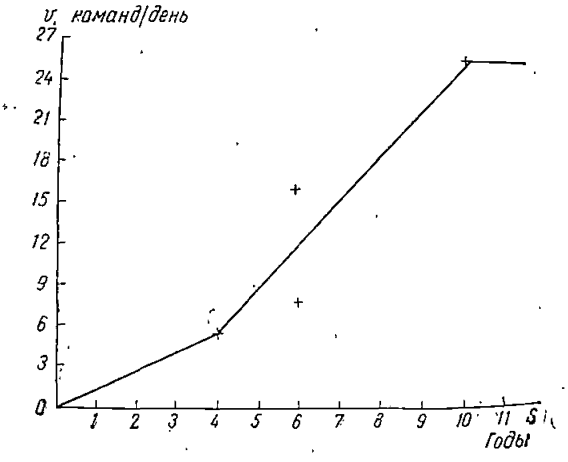



Рис. 2. Зависимость производительности труда программистов от стажа работы

грамм, производительность программиста при составлении практически не снижается.

При учете зависимости производительности от стажа работы выявляются три области (рис. 2). В первой области (при стаже работы до 4 лет) производительность изменяется по закону $V=1{,}2S$; вторая область (при стаже от 4 до 10 лет) характеризуется зависимостью $V=4{,}8+\frac{10}{3}(S-4)$. И наконец, в третьей области при стаже работы более 10 лет производительность практически не повышается и равна 25 командам в день. Такой характер зависимости можно объяснить, по нашему мнению, следующим образом. В начальный период овладения программированием (примерно до 3,5—4 лет) производительность возрастает медленно, причем, чем выше класс ЭВМ, тем, по-видимому, медленнее рост производительности. Затем при накоплении определенного объема знаний производительность программиста повышается быстрее. При стаже работы 10 и более лет программист больше уделяет внимания качеству программы, оптимизирует логику вычислений и т. д. В этот период не наблюдается роста производительности (в командах/день).

На основе полученной зависимости $V$ от $R$ (см. рис. 1) экспериментальные данные были приведены к производительности труда программиста с 6-летним стажем работы (табл. 2). Результаты вычислений были аппроксимированы степенной зависимостью, т. е. полиномом 2-го порядка $T=aR+bR^2$.

Таблица 2

| Шифр программиста | Размер составленной программы, команды | Фактическое время разработки программы, дни | Планируемое время разработки программ программистом с 6-летним стажем работы, дни |
|---|---|---|---|
| А | 388 | 15.7 | 25.6 |
| Б | 627 | 62.7 | 59.2 |
| В | 1249 | 160.0 | 164.0 |
| Г | 390 | 25.7 | 25.7 |
| Д | 567 | 72.8 | 50.1 |
| Е | 313 | 65.0 | 18.2 |
| Ж | 810 | 87.3 | 89.3 |
| З | 838 | 129.0 | 94.2 |
| И | 590 | 47.5 | 54.0 |

Используя метод наименьших квадратов, мы получили следующие оценки коэффициентов полинома:

$$a = 0.049 \text{ дня/команду} = 0.42 \text{ ч/команду}$$
$$b = 0{,}65 \cdot 10^{-4} \text{ дня/команду}^2 \simeq 6 \cdot 10^{-4} \text{ ч/команду}^2 \quad (1)$$

Данные оценки коэффициентов позволяют получать прогнозируемые значения времени разработки программы. Полученные нами данные несколько занижены по сравнению с оценками, приведенными в [2]: $a=1$ ч/команду, $b=1{,}6\cdot 10^{-4}$ ч/команду$^2$. Это объясняется тем, что из-за недостаточного для проведения оценок числа про-

граммистов мы использовали данные при разработке нескольких программ одним программистом. Это могло вызвать занижение значений, поскольку на практике благодаря последовательной разработке схожих программ повышается производительность при составлении программ, так как используются аналогичные блоки и приобретается навык в составлении программных решений.

Таким образом, полученную зависимость времени разработки программы от ее размера можно использовать при планировании загрузки программистов (рис. 3):

$$T = 0{,}049R + 0{,}65 \cdot 10^{-4}R^2, \qquad (2)$$

где $R$ — размер планируемой к разработке программы, команд;
$T$ — оценка времени разработки, дней.

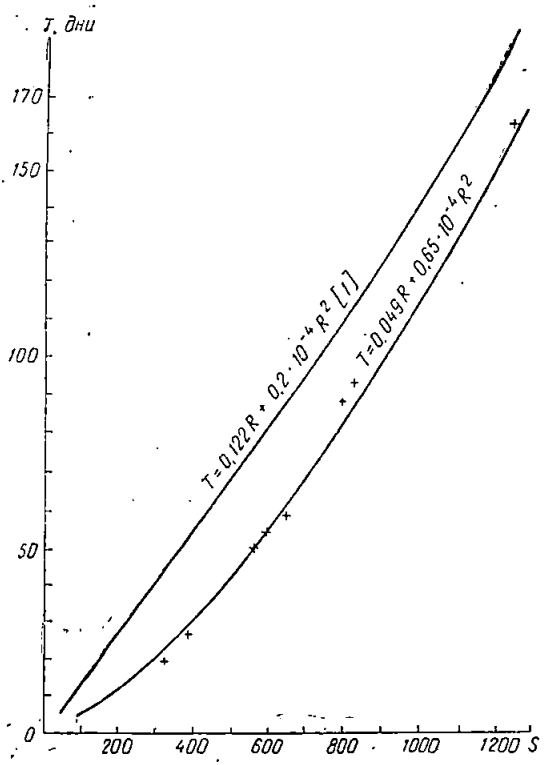



Рис. 3. Зависимость времени разработки программы от размера составляемой программы с учетом производительности программиста с 6-летним стажем работы

Для этого предварительно оценивается размер предполагаемой программы для решения задач по обработке экономической информации, а затем по формуле (2) вычисляется прогнозируемое время разработки программы. На основе разработанной библиотеки программных модулей решались задачи по формированию различных документов.

Экспериментальные замеры объемов работ (табл. 3) позволили получить оценки производительности труда программистов при разработке вычислительных алгоритмов с использованием библиотеки программных модулей.

Данные по программистам А, Д, Г показывают, что производительность существенно возросла (табл. 4). Снижение производительности труда программиста И объясняется следующими факторами: во-первых, стаж его работы небольшой, а во-вторых, в период разработки алгоритмов программист И одновременно изучал работу системы модульного программного обеспечения.

В заключение необходимо отметить, что нами представлены материалы первого опыта по оценке производительности труда программистов. Несомненно, получение нормативных оценок потребует детального изучения вопроса производительности работы программистов при составлении программ широкого класса.

Таблица 3

| Шифр программиста — составителя алгоритмов | Количество фраз в вычислительном алгоритме | Число использованных модулей |
|---|---|---|
| А | 90 | 17 |
| А | 62 | 11 |
| И | 36 | 6 |
| Д | 120 | 22 |
| Д | 120 | 22 |
| Д | 120 | 22 |
| Г | 88 | 10 |
| Д | 198 | 17 |
| Г | 191 | 13 |
| Г | 142 | 15 |
| Г | 142 | 15 |
| А | 584 | 17 |
| Г | 141 | 16 |

Таблица 4

| Шифр программиста | Объем составленных алгоритмов, м/команд | Число составленных алгоритмов, шт. | Период работы, дни | Число макрокоманд в день |
|---|---|---|---|---|
| А | 736 | 3 | 20 | 36.8 |
| Д | 558 | 4 | 40 | 13.92 |
| Г | 704 | 5 | 30 | 23.3 |
| И | 36 | 1 | 5 | 7.2 |

## ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

1. Васильев П. П. Модуль — математическое обеспечение системы обработки информации. — УСиМ, 1976, № 3.
2. Эрлих А. И., Вен В. Л. К вопросу о выборе структуры процесса программирования. — В кн.: Программный метод управления. ВЦ АН СССР, 1976, № 3.

*Н. О. ИВАНОВ, Е. И. МАКАРЕНКО,*
*С. И. СУРНОВ*

## КОНСТРУКТОР УПРАВЛЯЮЩИХ ПРОГРАММ

Эффективность использования ЭВМ существенно зависит от уровня специальной программистской подготовки пользователей. Поэтому целесообразно создание пакетов прикладных программ для

пользователей с недостаточным уровнем подготовки. Наибольший интерес представляют пакеты сложной структуры, обеспечивающие пользователю максимальные возможности и простоту работы.

Пакет сложной структуры состоит:

из интерпретатора входного языка пакета;

из управляющей программы;

из набора программных модулей.

Представляет определенный интерес задача создания некоторого алгоритма, позволяющего строить интерпретатор входного языка и управляющую программу пакета независимо от физического смысла программных модулей. Такой алгоритм можно трактовать как конструктор пакета. Проанализируем возможность его построения.

Задачами интерпретатора входного языка являются расшифровка программы пользователя, написанной на входном языке, и определение последовательности модулей, которые необходимо выполнить для решения поставленной пользователем задачи. Решение второй задачи зависит от заданных пользователем исходных данных и требуемого критерия решения задачи и в общем случае неоднозначно. Так как входные языки пакетов, предназначенных для решения различных классов задач, весьма специфичны, то и их интерпретаторы имеют ряд существенных особенностей. Кроме того, критерии, исходя из которых определяется последовательность модулей, зависят от класса задач, решаемых с помощью данного пакета, и могут быть весьма разнообразны. Таким образом, интерпретатор входного языка существенно зависит от специфики решаемых задач и поэтому его автоматическое конструирование затруднено.

Задачей управляющей программы является организация выполнения и взаимодействия модулей, указанных интерпретатором. При этом управляющая программа оперирует с именами модулей и данных, не привязываясь к их физическому смыслу, благодаря чему обеспечивается ее автоматическое конструирование. Более того, появляется возможность возложить на данную программу некоторые функции интерпретатора, в частности оценку выполнения синтезированной последовательности модулей. Отметим, что приобретенная функция становится и наиболее ответственной в управляющей программе, так как позволяет учесть степень задания исходных данных фактически на этапе синтеза последовательности модулей. Схема взаимодействия интерпретатора и управляющей программы показана на рис. 1.

Таким образом, принципиально возможно создание конструктора управляющих программ, входными данными которого являются: набор программных модулей, имена входных и выходных данных каждого модуля.

Очевидны и функции конструктора: упорядочение набора программных модулей, классификация входных данных пакета с целью оценки возможности выполнения синтезированной последовательности модулей.

Имена модулей образуют множество

$$M = \{m_i\}\ (i = 1,\ 2,\ \ldots,\ s), \tag{1}$$

где $s$ — число модулей в пакете.

Имена входных и выходных данных модуля $m_i$ образуют соответственно множества $X^i$ и $Y^i$. Множества имен входных и выходных данных всего пакета при этом определяются формулами

$$\left.\begin{aligned} X &= \bigcup_{i=1}^{s} X^i, \\ Y &= \bigcup_{i=1}^{s} Y^i, \end{aligned}\right\} \tag{2}$$

а общее множество имен данных пакета — формулой

$$D = X \cup Y. \tag{3}$$

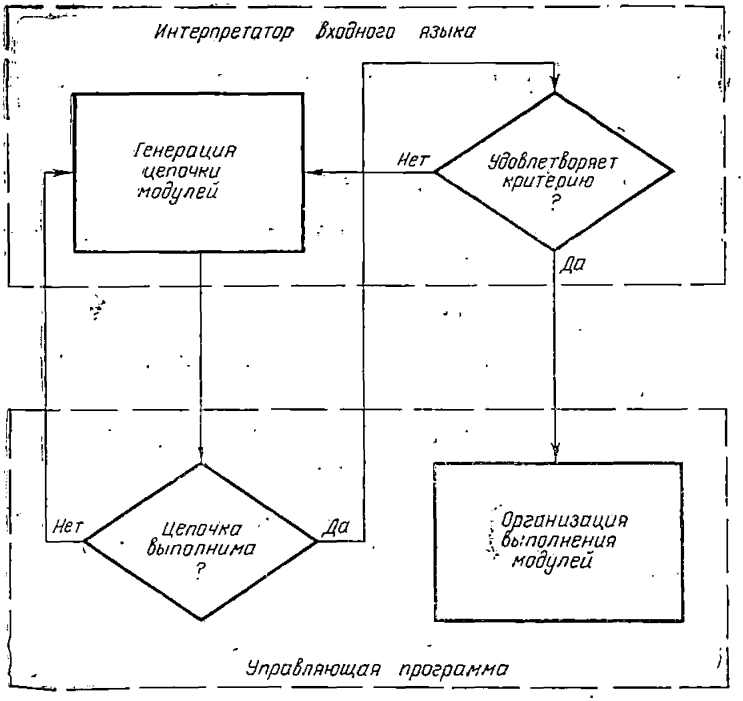



Рис. 1. Схема взаимодействия интерпретатора и управляющей программы

Отметим, что объединение модулей в пакет целесообразно только в том случае, если одни модули передают данные другим модулям, т. е. если существуют модули

$$m_l,\ m_k \in M$$

такие, что выполняется условие

$$X^l \cap Y^k \neq \varnothing \tag{4}$$

Создание пакета из модулей, для которых не выполняется условие (4), бессмысленно.

Конечной целью упорядочения набора программных модулей является построение ярусного графа, вершины которого — имена модулей, а дуги, соединяющие вершины, имеют смысл связей по данным. Эквивалентом графа является матрица смежности [2]

$$A = \{a_{ij}\}\ (i,\ j = 1,\ 2,\ \ldots,\ s),$$

где

$$a_{ij} = \begin{cases} 1, & \text{если } X^i \cap Y^j \neq \varnothing; \\ 0, & \text{если } X^i \cap Y^j = \varnothing. \end{cases} \tag{5}$$

Алгоритм построения графа состоит в следующем.

1. Для каждого $m_k \in M$ проверяется условие

$$X^k \cap Y^j = \varnothing (j = 1,\ 2,\ \ldots,\ s). \tag{6}$$

Если для модуля $m_k$ условие (6) выполняется для некоторого $j$, то

$$a_{kj} = 0.$$

Модули, удовлетворяющие условию (6), образуют множество $M^0 \subset M$.

2. Строятся множества $M^j$ исходя из следующего. Если для модуля $m_l \in M - M^0$ выполняется условие

$$X^l \cap Y^i_{j-1} \neq \varnothing, \tag{7}$$

где $Y^i{}_{j-1}$ — множество имен выходных данных модуля $m_i \in M^{j-1}$, то $m_l \in M^j$. При этом

$$a_{ll} = \begin{cases} 1, \text{ если для модуля } m_l \text{ выполняется условие (7)}; \\ 0 \text{ — в остальных случаях.} \end{cases}$$

Пункт 2 выполняется до тех пор, пока не будет найдено такое $k$, что $M^k = \varnothing$, т. е. для каждого $m_l \in M - M^0$ и для каждого $m_i \in M^{k-1}$ выполнится условие

$$X^l \cap Y^i_{k-1} = \varnothing. \tag{8}$$

Классификация входных данных пакета заключается в определении двух множеств:

$C$ — множества имен данных, задаваемых пользователем (данные первой группы);

$V$ — множества имен данных, значения которых могут быть получены в результате работы модулей (данные второй группы).

Общее множество входных данных пакета при этом определяется как

$$X = C \cup V. \tag{9}$$

Множество имен данных второй группы для каждого модуля может быть определено по следующим соотношениям:

$$\left.\begin{array}{l}\text{для } m_i \in M^0: \\ \quad V^i = 0; \\ \text{для } m_i \notin M^0: \\ V^i = \bigcup\limits_{j=1}^{s} (X^i \cap Y^j); \\ (i \neq j;\ a_{ij} = 1).\end{array}\right\} \qquad (10)$$

Множество имен данных первой группы определится из (9):

$$C^i = X^i - V^i. \qquad (11)$$

Упорядочение набора программных модулей и представление входных данных в виде множеств двух групп позволяют составить достаточно простой алгоритм управляющей программы. Для этого поставим множеству имен $D$ в соответствие множество значений $Z$ такое, что

$$Z \infty D. \qquad (12)$$

Тогда модуль $m_j$ в качестве исходных данных использует значения

$$Z^j_X \infty X^j, \qquad (13)$$

где $Z_{X^j} \subset Z$.

Отметим, что $Z_{X^j} = Z_{C^j} \cup Z_{V^j}$, где $Z_{C^j} \infty C^j$ и $Z_{V^j} \infty V^j$.

Пользователь для решения своей задачи задает некоторое множество значений $F$ такое, что $F \subset Z$. Результатом работы модуля $m_j$ является множество значений $Z_{Y^j} \infty Y^j$, где $Z_{Y^j} \subset Z$.

Алгоритм управления пакетом осуществляет следующие функции:

контроль возможности выполнения последовательности модулей, заданной вектором $P$;

организацию выполнения модулей.

При контроле проверяется наличие или своевременное получение исходных данных для всех модулей, имена которых содержатся в векторе $P$. В соответствии с проведенной классификацией данных контролируются обе группы.

Контроль данных первой группы: для каждого модуля с именем $p_i \in P$:

контроль 1 : if $Z_c^i \subset F$ then ($i := i + 1$; goto контроль 1)

else ошибка.

При контроле данных второй группы проверяется их наличие, а при отсутствии — их поступление от других модулей к моменту выполнения каждого модуля $p_i$. Для этого выполняются действия.

1. Моделируется выполнение каждого модуля $p_i$, т. е. уничтожаются связи (дуги), идущие от вершины графа, соответствующей модулю $p_i$:

$$a_{jl} = 0 \ (j = 1, 2, \ldots, s).$$

2. Осуществляется поиск:

$$M: \underline{\text{if}}\ a_{ij} = 1\ \underline{\text{then}}\ \underline{\text{goto}}\ \text{п. 4}\ \underline{\text{else}}\ (j := j + 1;\ \underline{\text{goto}}\ M).$$

3. Выбирается следующий модуль из вектора $P$ и осуществляется переход к п. 1.

4. Делается проверка:

$$\underline{\text{if}}\ (Z_Y^i \cap Z_x^i) \subset F\ \underline{\text{then}}\ \underline{\text{goto}}\ M\ \underline{\text{else}}\ \text{ошибка}.$$

После контроля данных организуется выполнение модулей: вызываются и активизируются модули с именами $p_i \in P$.

**Пример.**

Пусть $M = \{m_1, m_2, m_3\}$.

Входные и выходные данные модулей соответственно:

$$X^1 = \{a_1, b_1, c_1\}; \quad Y^1 = \{b_3\};$$
$$X^2 = \{a_2, b_2, c_2\}; \quad Y^2 = \{a_1, c_3\};$$
$$X^3 = \{a_3, b_3, c_3\}; \quad Y^3 = \{a_4, b_4\}.$$

Тогда множество выходных данных пакета определится как

$$Y = \{b_3, a_1, c_3, a_4, b_4\}.$$

В результате работы конструктора на первом этапе получим:

$$M^0 = \{m_2\};\ M^1 = \{m_1, m_3\};\ M^2 = \{m_3\}.$$

При этом матрица смежности имеет вид

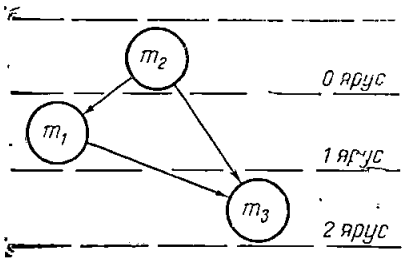



Рис. 2. Граф пакета, соответствующий полученной матрице смежности

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

Граф пакета, соответствующий полученной матрице смежности, показан на рис. 2.

Классификация входных данных пакета как результат второго этапа работы конструктора определяет множества:

для модуля $m_1$

$$V^1 = X^1 \cap Y^2 = \{a_1\};$$
$$C^1 = X^1 - V^1 = \{b_1, c_1\};$$

для модуля $m_2$

$$V^2 = \varnothing;$$
$$C^2 = X^2 - V^2 = \{a_2, b_2, c_2\};$$

для модуля $m_3$

$$V^3 = (X^3 \cap Y^1) \cup (X^3 \cap Y^2) = \{b_3,\ c_3\};$$

$$C^3 = X^3 - V^3 = \{a_3\}.$$

## ЛИТЕРАТУРА

1. Лебедев В. Н. Введение в системы программирования. М., Статистика, 1975.
2. Берзтисс А. Т. Структуры данных. М., Статистика, 1974.
3. Кахро М. И. и др. Система программирования Приз. — Программирование, 1976, № 1.

# 3 ЯЗЫКИ, СИСТЕМЫ ПРОГРАММИРОВАНИЯ, ОПЕРАЦИОННЫЕ СИСТЕМЫ

*Ю. П. МЕРКУШОВ*

## АНАЛИЗ НЕКОТОРЫХ ОСОБЕННОСТЕЙ ВЫДАЧИ ДИАГНОСТИЧЕСКИХ СООБЩЕНИЙ АССЕМБЛЕРОМ ОС ЕС ЭВМ

Система диагностических сообщений об ошибках в программах, написанных на Ассемблере, предоставляет возможность программисту достаточно эффективно обнаруживать и устранять синтаксические ошибки. Под синтаксическими ошибками в данном случае понимаются ошибки, связанные с неправильным написанием в программе машинных команд, команд Ассемблера и макрокоманд.

Информация о синтаксических ошибках выдается Ассемблером, как правило, на печать в следующем порядке:

в тексте объектной программы в листинге печатается индикатор ошибки в виде ***ERROR ***. Индикатор ошибки располагается на строке, следующей за предложением программы, в которой обнаружена ошибка;

в разделе листинга DIAGNOSTICS печатается список ошибок, обнаруженных в данной программе. Формат печати в списке ошибок имеет следующий вид:

nnn IEUXXX ТЕКСТ СООБЩЕНИЯ,

где nnn — номер предложения программы;
IEUXXX — идентификатор (IEU) и номер (XXX) диагностического сообщения;
ТЕКСТ СООБЩЕНИЯ содержит сведения о характере ошибки.

Однако опыт программирования на Ассемблере показал, что в некоторых случаях по выданным диагностическим сообщениям определить характер ошибки весьма трудно. В связи с этим было проведено моделирование различных ситуаций в программах, в которых предусматривается выдача диагностических сообщений о синтаксических ошибках. Анализ результатов моделирования показал, что диагностические сообщения IEU035, IEU059, IEU074, IEU101 и другие имеют специфические особенности выдачи на печать информации об ошибках.

Диагностическое сообщение IEU035 об ошибке в адресации выдается по вышеприведенным правилам в тех случаях, когда для использованного в данном предложении неявного адреса нет доступного регистра базы. Но это сообщение может быть выдано и тогда, когда в предложении программы, которому приписывается сообщение IEU035, ошибок нет. Ошибка в этом случае наблюдается в одном из предыдущих предложений данной программы, в котором неправильно определена явная адресная константа (константа типа S). Ошибка заключается в том, что в поле операндов этой константы записано выражение, для которого или нет доступного регистра базы, или значение выражения отрицательно. При этом неправильно определенная константа типа S в объектном модуле записывается в виде нулей, но ошибка не помечается в листинге индикатором ошибки. Индикатор ошибки и диагностическое сообщение IEU035 выдаются в этом случае только для первой машинной команды формата RX или RS, SI, SS, расположенной после (но необязательно непосредственно) константы типа S, определенной с ошибкой.

На рис. 1 и 2 показаны особенности выдачи диагностического сообщения IEU035. На рис. 1 представлены фрагменты транслированной программы на Ассемблере. При этом предполагается, что программа состоит только из двух программных секций PW1 и PW2. На рис. 2 дан список диагностических сообщений, соответствующий программе рис. 1. Из данных рисунков следует, что в соответствии с общими правилами в предложении 83 имеется ошибка — для неявного адреса C4 нет доступного регистра базы. Но в предложении программы 82 задан регистр базы, доступный для C4. Действительной причиной печати в данном месте программы индикатора ошибки и печати в списке ошибок диагностического сообщения IEU035 с привязкой к предложению программы 83 явилась запись в предложении 79 константы типа S. В операндах этой константы указан неявный адрес C5, для которого регистр базы в данном месте программы не определен. Если запись USING PW2, 7, сделанную в предложении 82, поместить до предложения 79, то данная ошибка будет устранена.

Необходимо отметить, что при использовании Тестрана для печати содержимого фиктивной области, определяемой оператором DSECT, печать диагностического сообщения IEU035 возможна и после 1-й машинной команды формата RR в отлаживаемой программе, хотя до включения команд Тестрана в программе ошибок не было. Печать информации об ошибках в данном случае объясняется тем, что в командах Тестрана не был определен регистр базы для фиктивной области программы, подлежащей распечатке.

Диагностическое сообщение IEU059 выдается в тех случаях, когда при генерации макрорасширения встретилось имя перехода, которое не было определено в макроопределении. В этом случае индикатор ошибки печатается после предложения, предшествующего генерируемому переходу, где указано неопределенное имя перехода. В списке ошибок данному диагностическому сообщению приписывается номер предложения, после которого напечатан ин-

дикатор ошибки в связи с неопределенным именем перехода. При этом предложение, которому приписано сообщение IEU059, может и не содержать ошибки.

```
LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT
001000                              1 PW1      START X'1000'
001000 0550                         2          BALR  5,0
001002                              3          USING *,5
001002 5860 5236          01238     4          L     6,A6
  ...  ...  ...            ...      ...        ...
001236 0000                         79 A5      DC    S(C5)
001238 4220000000000000             80 A6      DC    D'45'
001240                              81 PW2     CSECT
001240                              82         USING PW2,7
001240 0000 0000          00000     83         L     7,C4
  ***ERROR***
  ...  ...  ...            ....     ...        ...   ...
001300   00001240                   144 C4     DC    A(PW2)
001304                              145 C5     DS    E
001000                              146        END   PW1
```

Рис. 1. Фрагменты транслированной программы на Ассемблере

```
                    DIAGNOSTICS
STMT ERROR MESSAGE
  83 IEU035 NEAR OPERAND COLUMN 5 —— ADDRESSABILITY ERROR
   1 STATEMENT FLAGGED IN THIS ASSEMBLY
   8 WAS HIGHEST SEVERITY CODE
```

Рис. 2. Список диагностических сообщений, соответствующий программе рис. 1

Диагностическое сообщение IEU074 выдается в том случае, когда в макроопределении, вызываемом из макробиблиотеки на диске (набор данных с dd-именем SYSLIB), Ассемблер обнаруживает недопустимые команды. Предложение с недопустимой командой в листинге не печатается, а индикатор ошибки печатается. Следовательно, индикатор ошибки печатается вслед за предложением, которое предшествует ошибочному (хотя это предложение может и не иметь ошибок). В списке ошибок этому сообщению приписывается номер предложения, после которого напечатан индикатор ошибки в связи с использованием в макроопределении недопустимой команды. Кроме того, эта же ошибка приписывается и предложению END, которое также может не содержать ошибок. В данном случае в списке ошибок перед сообщением об ошибке в предложении END печатается примечание следующего содержания:

** ПРИМЕЧАНИЕ ** ОШИБКА МОГЛА БЫТЬ ОБНАРУЖЕНА ПРИ РЕДАКТИРОВАНИИ МАКРООПРЕДЕЛЕНИЯ ИЗ 'SYSLIB'

В связи с описанной выше трудностью обнаружения ошибок в макроопределениях необходимо перед их записью в макробиблиотеку проводить предварительную проверку на машине.

На рис. 3, 4 и 5 показан пример особенностей печати диагностических сообщений IEU059 и IEU074.

На рис. 3 показаны два макроопределения PR1 и PR2 в том виде, в каком они записаны в библиотеке. В макроопределении PR1 в операнде 1-го оператора AGO записано с ошибкой имя перехода .END вместо .MEND. В макроопределении PR2 записана недопустимая команда PRINT DATA.

```
        MACRO
        PR1     &R1,&R2
        AIF     (T'&R1 NE 'N').M1
        AIF     (T'&R2 NE 'N').M2
        AIF     (&R1 GT 15).M1
        AIF     (&R2 GT 15).M2
        MNOTE   *,'REAL OPERANDS'
        AR      &R1,&R2
        AGO     .END
.M1     MNOTE   10,'ILLEGAL OPERAND 1 — &R1'
        AGO     .MEND
.M2     MNOTE   10,'ILLEGAL OPERAND 2 — &R2'
.MEND   MEND
```

```
        MACRO
        PR2 &P1,&P2,&P3
        SR  3,3
        LA  4,&P1
        LD  2,&P2
        ME  2,(&P2+8)(3)
        LA  3,8(3)
        BCT 4,*—8
        PRINT DATA
        STE 2,&P3
        MEND
```

Рис. 3. Тексты макроопределения, записанные в библиотеку

На рис. 4 представлена печать текстов объектного и исходного модулей после трансляции, а на рис. 5 — печать списка диагностических сообщений, полученного по результатам трансляции программы, показанной на рис. 4.

```
LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT
001000                          1 BEG   START   X'1000'
001000 0550                     2       BALR    5,0
001002                          3       USING   *,5
001002 5860 5046        01048   4       L       6,C
                                5       PR2     3,A,B
001006 1B33                     6+      SR      3,3
001008 4140 0003        00003   7+      LA      4,3
00100C 6820 5026        01028   8+      LD      2,A
001010 7C23 502E        01030   9+      ME      2,(A+8)(3)
001014 4133 0008        00008  10+      LA      3,8(3)
001018 4640 500E        01010  11+      BCT     4,*—8
*** ERROR ***
00101C 7020 503E        01040  12+      STE     2,B
                               13       PR1     2,X'B'
                               14               *,REAL OPERANDS
001020 1A2B                    15+      AR      2,X'B'
*** ERROR ***
                               16               10,ILLEGAL OPERAND
                                                1—2
001022 000000000000
001028 4130000000000000        17 A     DC      D'3'
001030 4150000000000000        18       DC      D'5'
001038 4180000000000000        19       DC      D'8'
001040                         20 B     DS      D
001048 41F0000000000000        21 C     DC      D'15'
001000                         22       END     BEG
```

Рис. 4. Тексты объектного и исходного модулей

```
                        DIAGNOSTICS
STMT ERROR CODE   MESSAGE
  11 IEU074       ILLEGAL STATEMENT IN COPY CODE OR SYSTEM
                  MACRO
  15 IEU059       UNDEFINED SEQUENCE SYMBOL
  16 IEU037       MNOTE STATEMENT
  ** NOTE **      ERRORS MAY HAVE BEEN DETECTED WHILE
                  EDITING 'SYSLIB' MACROS
  22 IEU074       ILLEGAL STATEMENT IN COPY CODE OR SYSTEM
                  MACRO
   4 STATEMENTS   FLAGGED IN THIS ASSEMBLY
  12 WAS HIGHESTSEVERITY CODE
```

Рис. 5. Список диагностических сообщений, полученный по результатам трансляции программы, показанной на рис. 4

Анализ рис. 4 и 5 показывает следующее.

1. Если руководствоваться общими правилами расшифровки информации об ошибках, то тогда следует, что в предложении 11 имеется ошибка. Этой ошибке соответствует диагностическое сообщение IEU074 — в данном предложении записана недопустимая команда Ассемблера. На самом же деле недопустимая команда — PRINT DATA записана в предложении макроопределения PR2, которое следует за машинной командой BCT. Сообщение IEU074 фиксирует ошибку не в предложении 11, а в макроопределении PR2, ранее записанном с ошибкой в макробиблиотеку. По этой же причине предложению 22 (END), в данном случае также не имеющему ошибок, приписано диагностическое сообщение IEU074. Перед этим сообщением в списке ошибок на рис. 5 напечатано соответствующее примечание.

2. Предложению 15 приписаны индикатор ошибки и диагностическое сообщение IEU059. В соответствии с общими правилами эта информация расшифровывается, как запись в предложении 15 неопределенного имени перехода. В действительности же в предложении 15 ошибок нет. Ошибка имеется в макроопределении PR1, где в поле операндов 1-й команды AGO записано неопределенное имя перехода .END. Ошибка, связанная с записью в макроопределении неопределенного имени перехода, может привести к ошибочной печати предложения MNOTE. Так, на рис. 4 предложение 16, свидетельствующее об ошибке в 1-м операнде макрокоманды PR1, выдано ошибочно — 1-й операнд этой макрокоманды имеет допустимое значение.

Диагностическое сообщение IEU101 выдается в тех случаях, когда перед перфокартой с управляющим оператором /* (оператор ограничения — конец данных) не было перфокарты с командой END. Индикатор ошибки в этом случае печатается вслед за последним предложением программы, после которого должна следовать команда END, и номер этого предложения приписывается в списке ошибок диагностическому сообщению IEU101.

К числу других случаев, имеющих особенности выдачи на печать информации о синтаксических ошибках, относятся случаи, когда эта информация выдается без печати индикатора ошибки

и номера предложения программы в списке ошибок, к которому отнесено данное диагностическое сообщение. Эти особенности печати информации об ошибках наблюдаются в предупреждающих сообщениях и сообщениях об ошибках в управляющих операторах.

Под предупреждающими сообщениями будем понимать такие диагностические сообщения, при которых внимание программиста акцентируется на то, что при исполнении программы может возникнуть ошибка. К таким сообщениям относятся: IEU046, IEU058, IEU114. Так, сообщение IEU114 предупреждает о том, что в программе использованы машинные команды или константы, являющиеся некорректными для моделей ЕС-1020, ЕС-1022, ЕС-1030, ЕС-1040 или ЕС-1050. К числу таких команд относятся команды, приведенные в таблице.

| Команда | Мнемоника | Формат |
|---|---|---|
| ПЕРЕСЫЛКА ДЛИННАЯ | MVCL | RR |
| СРАВНЕНИЕ КОДОВ ДЛИННОЕ | CLCL | RR |
| ЗАГРУЗКА ПОЛНАЯ ДЛИННАЯ | LRDR | RR |
| ЗАГРУЗКА ПОЛНАЯ КОРОТКАЯ | LRER | RR |
| УМНОЖЕНИЕ РАСШИРЕННОЕ | MXR | RR |
| УМНОЖЕНИЕ РАСШИРЕННОЕ ДЛИННОЕ | MXDR | RR |
| СЛОЖЕНИЕ РАСШИРЕННОЕ С НОРМАЛИЗАЦИЕЙ | AXR | RR |
| ВЫЧИТАНИЕ РАСШИРЕННОЕ С НОРМАЛИЗАЦИЕЙ | SXR | RR |
| УМНОЖЕНИЕ РАСШИРЕННОЕ ДЛИННОЕ | MXD | RX |
| СРАВНЕНИЕ КОДА С СИМВОЛОМ ПО МАСКЕ | CLM | SI |
| ЗАПИСЬ СИМВОЛА В ПАМЯТЬ ПО МАСКЕ | STCM | SI |
| ЧТЕНИЕ СИМВОЛА ПО МАСКЕ | ICM | SI |

К числу констант, являющихся некорректными для указанных моделей ЕС, относится константа типа L, которая определяет расширенное число с плавающей точкой (два двойных слова).

Сообщения об ошибках в управляющих операторах языка управления заданиями выдаются Ассемблером в связи с несоответствием варианта работы, указанного в операндах управляющего оператора EXEC, с описанием данных (сообщения IEU111, IEU112, IEU113) или в связи с неправильным указанием варианта работы Ассемблера (сообщения IEU055, IEU078).

Описанные в данной статье особенности выдачи диагностических сообщений относятся к операционным системам ОС 4.0 и ОС 4.1 и ОС 1.0, за исключением того, что сообщения IEU114 в ОС 1.0 не выдаются.

Таким образом, знание приведенных в данной статье особенностей выдачи Ассемблером информации о синтаксических ошибках упростит в описанных случаях сложную и трудоемкую работу по отладке программ.

*Е. А. АЛЕКСЕЕНКО, А. М. ДОВГЯЛЛО,*
*О. П. НЕБРАТ, Б. А. ПЛАТОНОВ,*
*А. А. СТОГНИЙ*

# СИСТЕМА ПРОГРАММИРОВАНИЯ И ПОДДЕРЖАНИЯ ОБСЛУЖИВАЮЩИХ И ОБУЧАЮЩИХ КУРСОВ

Эффективное взаимодействие человека с вычислительной машиной требует развития программного обеспечения ЭВМ в области создания «обслуживающей и обучающей среды» [1, 2, 3], которая окажет следующую помощь пользователю:

а) сообщит необходимые сведения об интересующей его задаче, возможных средствах ее решения;

б) поможет уточнить формулировку задачи, разработать алгоритм ее решения, построить программу, не изучая детально язык программирования;

в) создаст эффективную подготовку к применению машинных средств для решения задач.

Если, например, пакету прикладных программ придать обучающую программу, обеспечивающую автоматизированную подготовку потенциальных пользователей данного пакета, то благодаря этому:

облегчается и ускоряется эксплуатация и внедрение пакетов;

разработчики пакетов во многом освобождаются от подготовки пользователей, обеспечивая только знакомство и консультации по тому материалу, который не может быть усвоен с помощью обучающей программы;

облегчается работа подразделения сопровождения, которое может не вникать во все тонкости конкретного программного продукта, обеспечивая только его содержание в соответствии с инструкцией по эксплуатации.

В условиях постоянного развития информационных и алгоритмических возможностей ЭВМ помощь и обучение, осуществляемые машиной, необходимы не только для «новичков» и пользователей-непрофессионалов, но и для квалифицированных программистов, желающих эффективно применять в своей работе новые программы и данные.

Наконец, в плане общеобразовательной подготовки кадров создание автоматизированных обучающих, имитационных и тренирующих систем на базе ЭВМ становится одним из главных направлений, обеспечивающих эффективную профессиональную подготовку специалистов различного профиля, студентов вузов и техникумов, учащихся средних школ [4, 5].

Работы в данной области уже вышли за рамки экспериментов как у нас в стране, так и за рубежом. В частности, в США за минувшее десятилетие обучающие системы на базе ЭВМ прошли путь развития от частных реализаций, работающих с двумя-тремя пользователями по одному-двум предметам, до вычислительных сетей учебного назначения, обслуживающих сотни удаленных терминалов

и располагающих банками обучающих программ самого различного назначения. Примером такой сети является система PLATO-IV фирмы CDC [6], автоматизированные обучающие системы фирм IBM, DEC, Hewlett — Packard.

С появлением машин Единой системы возникли реальные предпосылки к созданию и внедрению растущей обучающей и обслуживающей «среды» и у нас в стране. Рассматриваемая в данной статье Система Программирования и поддержания обслуживающих и Обучающих Курсов (СПОК) [7, 8] является первой в СССР специализированной системой, работающей в режиме разделения времени и позволяющей:

применять ЭВМ для оказания помощи, обучения и контроля знаний десятков пользователей, работающих на машине одновременно с индивидуальными диалоговыми программами;

существенно ускорять и облегчать процесс создания прикладных обслуживающих и обучающих диалоговых программ (курсов) для разнообразных применений;

обеспечивать администрацию вычислительного центра или учебного заведения необходимыми данными о ходе и результатах автоматизированного обучения и обслуживания пользователей.

## ВЗАИМОДЕЙСТВИЕ ПОЛЬЗОВАТЕЛЕЙ И СИСТЕМЫ

СПОК обслуживают четыре категории пользователей: диспетчера, авторов прикладных программ, преподавателей и обучаемых. Каждая категория пользователей имеет в системе свои языковые средства и взаимодействует с ней в режиме диалога.

Подготовка системы к работе и связь с ней пользователей осуществляются следующим образом. Вначале оператор ЭВМ, опираясь на имеющуюся конфигурацию технического обеспечения машины ЕС ЭВМ, по специальной инструкции, входящей в документацию СПОК, осуществляет генерацию программ пакета. Затем диспетчер, являющийся администратором системы, входит в контакт с ней, регистрирует в системе прикладные диалоговые программы (курсы), авторов, которые их будут создавать, и обучаемых-пользователей прикладных программ. Далее, в процессе функционирования системы диспетчер следит за правильностью ее работы, эффективностью использования прикладных программ и ресурсов вычислительной машины.

После завершения регистрации к работе с системой могут приступать авторы — специалисты той предметной области, по которой пишется диалоговый курс. В их задачи входят методическая проработка содержания курса, разработка сценария и программы диалога, написание программы на Языке Описания диалоговых Курсов (ЯОК)*, ввод ее в библиотеку СПОК и отладка введенной программы. ЯОК прост в изучении и доступен авторам, не имеющим специальной подготовки в области программирования и вычислительной техники.

* ЯОК — аналог языка COURSEWRITER III [15].

Возможные расширения ЯОК позволяют включить в него подпрограммы, написанные на языках Ассемблер, ПЛ/1, Кобол и Фортран.

Преподаватели или консультанты помогают пользователям (обучаемым) в их работе с диалоговыми курсами. В частности, используя статистические данные о ходе процесса обучения, выводимые на терминал, преподаватели могут активно вмешиваться в процесс обучения. Следует отметить, что присутствие преподавателей при работе обучаемых с прикладными программами необязательно.

СПОК обеспечивает эффективное программирование и поддержание следующих основных режимов взаимодействия с пользователями прикладных программ:

консультацию или справку (типа «меню» [10]);

программированное обучение [11];

обучающе-справочный режим;

диагностику потребностей и/или знаний пользователя (обучаемого) с последующей отсылкой его к соответствующим печатным материалам;

контроль знаний, умений и навыков с выставлением оценки и распечаткой протокола контроля;

«беседу» с целью обучения или диагноза;

практическую работу с изучаемыми пакетами прикладных программ, системами программирования и др.;

управляемое машиной программирование на незнакомом пользователю языке [3, 12].

## СИСТЕМА ПРОГРАММИРОВАНИЯ ОБСЛУЖИВАЮЩИХ И ОБУЧАЮЩИХ КУРСОВ КАК СИСТЕМОЛОГИЧЕСКИЙ ОБЪЕКТ

Анализ эффективного функционирования программных систем, ориентированных на создание и поддержание обучающих программ [4, 5], приводит к необходимости применения при разработке таких систем ряда положений системологии [9]. В частности, при разработке СПОК учитывались следующие факторы:

1. Работоспособность системы в широком спектре одновременно изменяющихся воздействий внешней среды. Внешней средой для СПОК являются четыре категории пользователей: пользователи прикладных программ СПОК; авторы этих прикладных программ; консультанты (преподаватели) и диспетчер-администратор системы.

2. Возможность замены технических компонентов системы. СПОК может функционировать на любой машине, реализующей язык Ассемблер (ЕС ЭВМ, начиная с ЕС-1020, М-4030), с различными конфигурациями стандартного терминального оборудования Единой системы.

3. Многофункциональность. В СПОК — это внешние и внутренние цели. Внешними являются цели, связанные с обеспечением некоторого показателя качества обслуживания и/или обучения пользователей курсов. Средства достижения внешних целей заложены

в самих курсах. Когда этих средств недостаточно, в процесс достижения внешней цели предусматривается включение преподавателя-консультанта.

Внутренними целями являются обеспечение минимального времени реакции СПОК на введенное сообщение, контроль правильности работы, сохранение целостности системы, а также идентификация и переработка воздействий среды.

4. Гибкость при корректировке и расширении системы, т. е. возможность внесения изменений несколькими способами без существенных затрат времени и средств. В СПОК это требование реализуется благодаря простой возможности расширения языка описания курсов специальными операторами, необходимыми авторам прикладных программ, а также благодаря независимой корректировке программных модулей системы.

5. Простота взаимодействия с внешней средой. Обеспечение этого условия в СПОК достигается за счет проблемной ориентации входного языка и максимального приближения его к естественному языку данной предметной области. Входной язык СПОК состоит из двух основных подмножеств: специализированного языка программирования (ЯОК) и языков директив указанных выше категорий пользователей.

6. Возможность «обучения». Под «обучением» [9] СПОК понимается процесс создания внешней средой с помощью входного языка системы некоторой прикладной программы с целью придания системе требуемой функциональной ориентации (рис. 1).

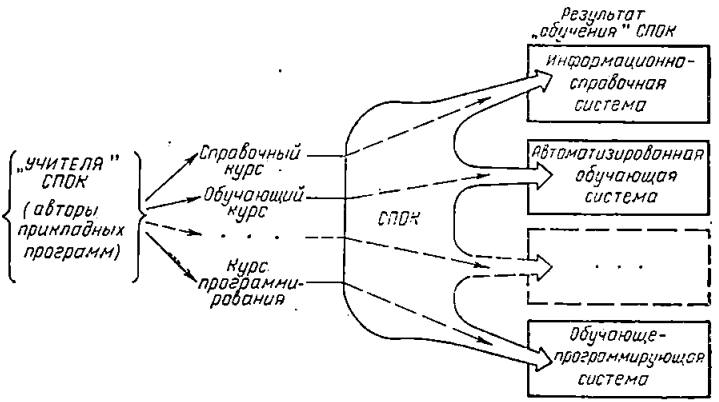



Рис. 1. «Обучение» СПОК

## АРХИТЕКТУРА СПОК

СПОК является пакетом прикладных программ специального назначения и представляет собой диалоговую систему реального времени, реализующую для обслуживания пользователей режим разделения времени.

Для работы СПОК необходима версия ДОС ЕС, включающая программы базисного телекоммуникационного метода доступа ВТАМ. СПОК может занимать любой из разделов ДОС ЕС.

Как всякий пакет прикладных программ ЕС ЭВМ [13] СПОК выполняет собственные функции после этапов генерации и инициализации.

СПОК состоит из технических и программных компонентов, а также из эксплуатационной документации для каждой из категорий пользователей.

### ТЕХНИЧЕСКИЕ КОМПОНЕНТЫ СПОК

Одна из возможных конфигураций технических компонентов СПОК приведена на рис. 2. Все показанные на рисунке устройства обязательны при функционировании СПОК, исключение составляет устройство SYSRDR (ЕС-5052, ЕС-5017 или ЕС-6012), которое закрепляется за пакетом СПОК только на момент ввода задания инициализации пакета, а затем это назначение может отменяться.

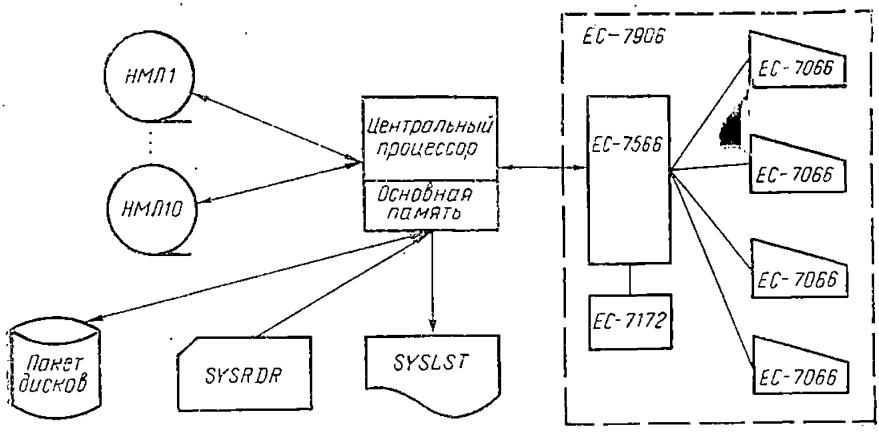



Рис. 2. Конфигурация технического обеспечения СПОК

Конкретная конфигурация технического обеспечения СПОК задается на этапе генерации оператором той ЭВМ ЕС, на которой пакет будет работать.

В конфигурацию может быть включено:

от 1 до 10 накопителей на магнитной ленте;

от 3 до 12 файловых экстентов, которые могут располагаться на одном или нескольких пакетах магнитных дисков;

от 1 до 5 устройств управления ЕС-7566 комплексов ЕС-7906, т. е. число одновременно работающих терминалов пользователей может колебаться от 1 до 80 на моделях ЕС-1020, ЕС-1022, ЕС-1030 и ЕС-1033, имеющих один мультиплексный канал.

Объем оперативной памяти, необходимой для функционирования пакета, зависит, в частности, от конфигурации технического обеспечения, но всегда должен быть больше 45 Кбайт. За пакетом СПОК должен быть постоянно закреплен таймер.

### ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СПОК

Подсистемы, составляющие программное обеспечение пакета, работают на разных этапах его функционирования, различным образом организованы и физически располагаются на различных устройствах (рис. 3).

Соответствующие части программного обеспечения СПОК (исходные, объектные или абсолютные модули) могут быть каталогизированы либо в соответствующие библиотеки резидентного пакета ДОС, либо в специально созданные библиотеки СПОК.

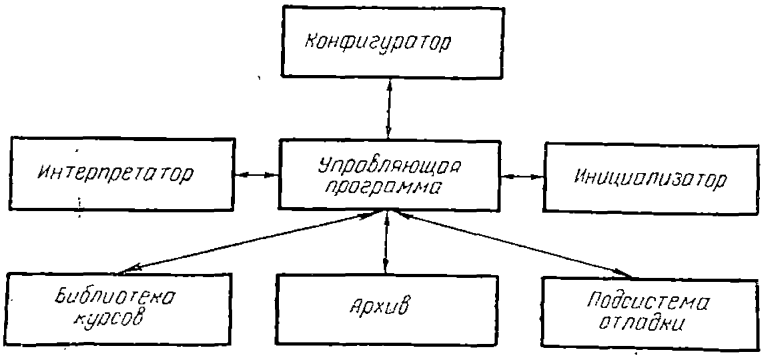



Рис. 3. Структура программного обеспечения СПОК

**Конфигуратор** представляет собой набор макроопределений, реализованных в языке Ассемблер, и хранящихся в библиотеке исходных книг пакета СПОК. Следует оговориться, что среди файловых экстентов, упомянутых в качестве возможных в описании конфигурации технических компонентов СПОК, экстенты для библиотек SL, RL и CL не предусматривались.

Макроопределения конфигуратора используются на этапе генерации пакета СПОК, обеспечивая адаптацию СПОК к различным конфигурациям технического обеспечения системы.

Макрокоманды, с помощью которых достигается генерация, описываются ниже в форматах, принятых в документации по ЕС ЭВМ. Параметры, используемые в макрокомандах, являются ключевыми.

**Макрокоманда CANAL** предназначена для задания конфигурации терминального оборудования СПОК. Формат этой макрокоманды выглядит следующим образом:

| Имя | Операция | Операнды |
|---|---|---|
| | CANAL | [BASE= {SYS016 / SYSnnn}], TYPE=(уу,трм,рэд,кол [,уу,трм,рэд,кол]...) |

Здесь BASE — определяет начальный номер логических устройств терминальной сети СПОК. По умолчанию этот номер равен 016, в противном случае nnn;

TYPE — определяет конфигурацию локальных дисплеев;

уу — тип устройства управления, может быть 7566;

трм — тип терминала, может быть 7066;

рэд — размер экрана дисплея, задается буквами А, В, С и Д: А задает размер экрана в 12×80=960 байт (12 строк по 80 байт); В — 12×40=480; С — 6×80=480 и Д — 6×40=240 байт;

кол — количество терминалов на устройстве управления. Задается десятичным числом от 1 до 32.

**Макрокоманда TAPES** предназначена для описания файлов на магнитных лентах, используемых СПОК. Формат макрокоманды следующий:

| Имя | Операция | Операнды |
|---|---|---|
| | TAPES | [BASE= {SYS010 / SYSnnn}] [,TYPE= {5012,2 / тип,кол}] [,ASSGN= {SYS010 / SYSnnn}] [,ALT=альтернатива] |

Где BASE — определяет начальный номер логических устройств пакета СПОК для файлов на магнитной ленте;

TYPE — определяет тип накопителя и их число;

тип — определяет тип НМЛ, может быть 5012;

кол — задает количество файлов, может быть 2;

ASSGN — определяет номер логического устройства, на которое службой архива осуществляется запись статистических данных о работе пользователей прикладных программ СПОК;

ALT — определяет номер логического устройства, на которое при заполнении устройства, заданного в параметре ASSGN, будет продолжать писаться статистическая информация; формат: SYSnnn, SYSnnn+1, где SYSnnn+1 — дополнительное устройство к SYSnnn.

**Макрокоманда DISKS** используется для описания файлов прямого доступа. Формат макрокоманды приведен ниже.

| Имя | Операция | Операнды |
|---|---|---|
| | DISKS | [BASE= {SYS020 / SYSnnn}] [,TYPE= {(5056,512,2) / (тип,дз,кф)}] [,SUBR= {(5056,2048) / (тип,дз)}] [,LDA= {(5056,2048) / (тип,дз)}] |

Где BASE — определяет начальный номер логических устройств пакета СПОК для файлов прямого доступа, хранящих программы курсов;

TYPE — определяет тип устройства прямого доступа, длину записи файла программ курса и количество этих файлов на томе (не более 10 файлов);

тип — определяет тип НМД, может быть 5056;

дз — длина записи в байтах, может быть 512;

кф — количество файлов, задается десятичным числом от 1 до 10;

SUBR — определяет тип устройства прямого доступа и длину записи для файла нерезидентных подпрограмм СПОК;

тип — определяет тип НМД, может быть 5056;

дз — длина записи в байтах, может быть 2048;

LDA — определяет тип устройства прямого доступа и длину записи для файла областей данных линии (ОДЛ) СПОК;

тип — определяет тип НМД, может быть 5056;

дз — длина записей в байтах, может быть 2048.

**Макрокоманда CORE** предназначена для описания рабочих полей СПОК в основной памяти ЭВМ. Формат макрокоманды следующий:

| Имя | Операция | Операнды |
|---|---|---|
| | CORE | [LDAS= {2/n}] [,SUBRS= {4/n}] [,TPBUF= {10/n}] |

Здесь LDAS — определяет число 2048 байтовых полей в оперативной памяти для размещения областей данных линии;

SUBRS — определяет число 2048 байтовых полей для размещения в оперативной памяти нерезидентных подпрограмм;

TPBUF — определяет число 120 байтовых буферов, из которых в СПОК при операциях вывода на терминал может быть организован буферный пул.

Использование при генерации макрокоманды CORE с различными значениями параметров позволяет добиться приемлемого времени реакции СПОК на сообщения пользователей при одновременном управлении объемом занимаемой основной памяти. Чем больше программ СПОК будет находиться в основной памяти, тем меньше будет реакция системы. Таким образом, макрокоманда CORE является одним из средств задания внутренней цели функционирования СПОК.

**Использование макроопределений ДОС.** При генерации пакета СПОК могут быть использованы макрокоманды ДОС EC-DAMOD, MTMOD, PRMOD и BTMOD со значениями параметров, задающими те или иные условия выполнения программ методов доступа при функционировании СПОК. Если эти макрокоманды при генерации пакета не были использованы, программы методов доступа включаются в программное обеспечение СПОК на этапе редактирования связей программ.

Результатом работы конфигуратора являются несколько объектных модулей, которые на этапе редактирования связей программ включаются в программную фазу, представляющую собой резидент СПОК.

**Подсистема или программа отладки** вызывается в основную память и начинает работать в случае обнаружения неустранимых программных сбоев в процессе функционирования СПОК, обеспечивая достижение одной из внутренних целей системы — контроля за правильностью работы.

Программа отладки осуществляет дампинг памяти раздела, в котором работала СПОК, на устройство SYSLST (см. рис. 2).

Тот или другой вид выполнения программы отладки может быть задан на этапе генерации СПОК макрокомандой, формат которой приведен ниже:

| Имя | Операция | Операнды |
|---|---|---|
| | DBUG | $\left[\text{FORMDMP}=\left\{\frac{\text{NO}}{\text{YES}}\right\}\right]$ |

Здесь FORMDMP — ключевое слово параметра, задающего тип дампинга;

NO — определяет включение в программу отладки подпрограммы форматизации дампинга ДОС ЕС;

YES — определяет включение в программу отладки собственной программы управления форматизации дампингов.

**Архив** в СПОК служит для накопления и обработки статистических данных о работе пользователей прикладных программ. Эти данные размещаются в однотомном или многотомном файле архи-

ва на магнитной ленте, что задается при генерации системы параметром ALT в макрокоманде TAPES.

Записи о работе пользователей (в частности, обучаемых) помещаются в файл архива по мере их возникновения в системе друг за другом. В связи с этим в пакете СПОК имеется специальная утилита для преобразования и вывода на устройство SYSLST записей из файла архива в заданном виде.

Вид накапливаемой в архиве информации задается диспетчером СПОК по заявке консультанта (преподавателя) и может быть использован последним для достижения внешних целей СПОК.

**Библиотека курсов** прикладных программ СПОК организована на магнитном диске в виде файлов прямого доступа. Число этих файлов может быть переменным и определяется при генерации пакета СПОК операндами BASE и TYPE макрокоманды DISKS.

Перед выполнением генерации пакета экстенты соответствующих файлов курсов утилитой CLRDSK должны быть очищены нулями и форматизованы. Длина записи в файлах составляет 512 байт.

Доступ в библиотеку курсов осуществляется системой по имени курса, которое задается диспетчером при регистрации курса. В каждом файле может быть зарегистрировано до 42 курсов, таким образом, при максимальном числе файлов (10) в СПОК может быть зарегистрировано 420 курсов. Библиотека курсов не может быть многотомной.

После регистрации курса пользователи обращаются к нему только по имени, не называя при этом ни экстента, на котором расположен курс, ни номера файла.

Программы из файлов курсов в оперативную память вызываются блоками по 512 байт и размещаются в соответствующих полях областей данных линии, определенных параметром LDAS макрокоманды CORE.

**Инициализатор.** Выполнение этапа инициализации возложено на программу-инициализатор, загружаемую в оперативную память.

Инициализатор выполняет две функции: самоконтроль СПОК и подготовку СПОК к работе. Самоконтроль СПОК заключается в проверке соответствия типов терминалов, указанных при генерации СПОК, типам назначаемых в задании инициализации пакета физических устройств. Это осуществляется путем анализа строк таблицы логических устройств (LAB) и соответствующих строк таблицы физических устройств (PAB), которые расположены в супервизоре ДОС ЕС.

Подготовка СПОК к работе состоит в следующем. По ряду причин заказанное на этапе генерации максимальное число терминалов не всегда будет использоваться. В связи с этим могут быть освобождены и соответствующие зоны основной памяти или блоки управления (БУ), содержащие информацию, необходимую для обработки сообщений данного терминала. При подготовке СПОК к работе инициализатор выполняет реорганизацию очереди БУ. На рис. 4 приведен алгоритм работы этой программы.

**Интерпретатор** осуществляет интерпретацию входного языка СПОК — языков директив всех категорий пользователей, а также операторов языка описания курсов.

Подпрограммы, составляющие интерпретатор, оформлены как нерезидентные, т. е. предназначены для хранения на магнитном диске. Файл прямого доступа, в котором они хранятся, определя-

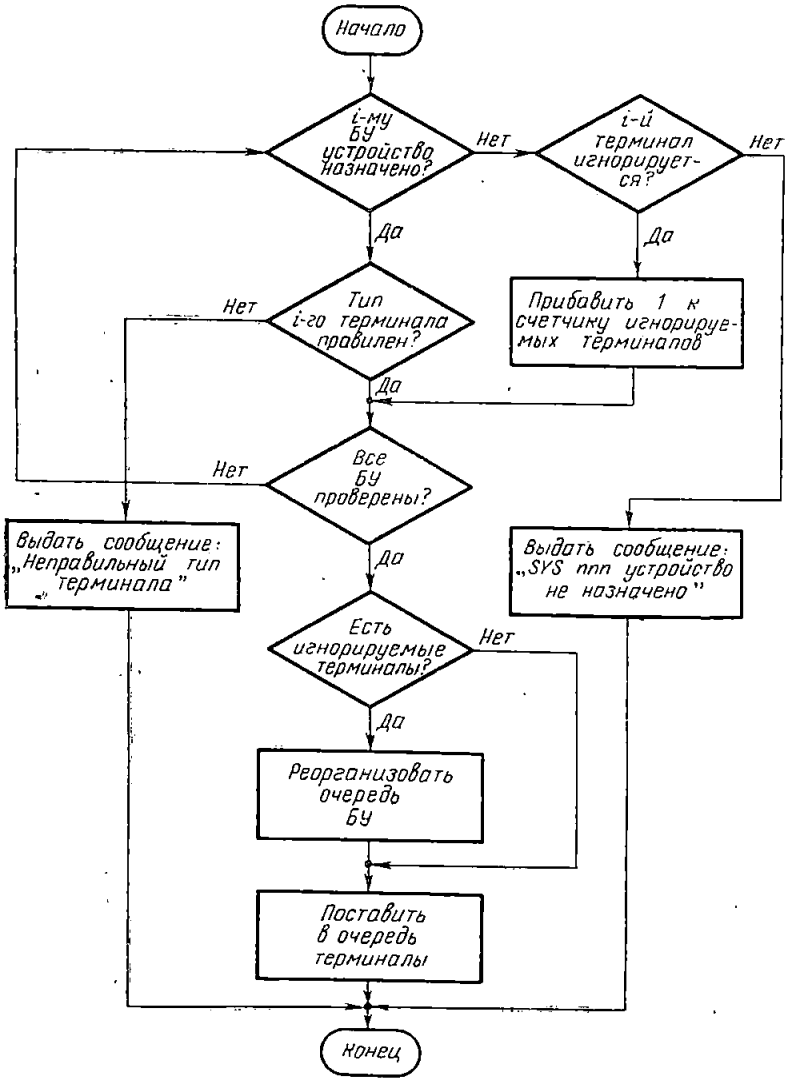



Рис. 4. Алгоритм работы программы инициализации

ется при генерации СПОК в параметре SUBR макрокоманды DISKS. Перед генерацией этот файл утилитой CLRDSK должен очищаться нулями и приводиться к длине записи 2048 байт.

При необходимости каждая из подпрограмм интерпретатора может быть считана с диска и размещена в область рабочих полей интерпретатора (РПИ), определенную параметром SUBRS макрокоманды CORE при генерации СПОК.

Для достижения внутренней цели функционирования СПОК (минимизации времени реакции при одновременном управлении объемом занимаемой основной памяти) в системе реализован механизм динамического перевода некоторых подпрограмм из статуса нерезидентных в статус «квазирезидентных», т. е. таких, которые сохраняются в основной памяти достаточно длительное время. Общее число нерезидентных и квазирезидентных подпрограмм задается значением параметра SUBRS макрокоманды CORE.

Способ минимизации времени реакции основан на подсчете количества обращений к той или иной подпрограмме интерпретатора, считанной с диска и находящейся в РПИ. Если при всех занятых в основной памяти РПИ возникает необходимость считать с диска еще одну подпрограмму, то она читается в РПИ на место той подпрограммы, к которой было наименьшее число обращений. Таким образом, подпрограммы, к которым обращаются часто, динамически переходят в разряд квазирезидентных, обеспечивая тем самым минимизацию среднего времени реакции.

В общем виде обработка интерпретатором директив пользователей осуществляется следующим образом. В зависимости от типа пользователей (автора, обучаемого и т. п.) вызывается соответствующая подпрограмма интерпретатора, распознающая директиву и определяющая ее корректность. Если в директиве имеется ошибка, пользователю выдается диагностическое сообщение о ней и осуществляется переход к ожиданию следующей директивы. Если ошибки нет, то вызывается подпрограмма, отрабатывающая данную директиву. Таких подпрограмм может быть несколько. После отработки директивы пользователю сообщается о готовности принять следующую директиву.

Несколько по-иному отрабатываются операторы ЯОК, вводимые автором. Если после проверки корректности оператора ошибки не обнаружено, то оператор трансформируется к виду, удобному для последующего исполнения, и помещается в библиотеку курсов в область, соответствующую курсу, с которым автор работает, и осуществляется переход в режим ожидания следующего оператора.

При ошибках в операторе ЯОК выдается соответствующее диагностическое сообщение, а сам оператор игнорируется.

**Управляющая программа** организует выполнение собственных работ пакета СПОК. Физически она располагается в основной памяти раздела, в котором функционирует система. На рис. 5 показана функционально-структурная схема СПОК, позволяющая описать основные функции управляющей программы и функционирование пакета. Все указанные на схеме и находящиеся в основной

памяти (ОП) компоненты относятся к объектам трех типов: данным, рабочим полям и программам.

К данным, которыми пользуется управляющая программа, относятся блоки управления и области данных линии.

Как уже упоминалось, БУ содержат всю необходимую информацию, относящуюся к обрабатываемому сообщению. В частности, в БУ запоминается само сообщение и в течение его прохождения через систему любые необходимые отметки или данные, относящиеся к нему.

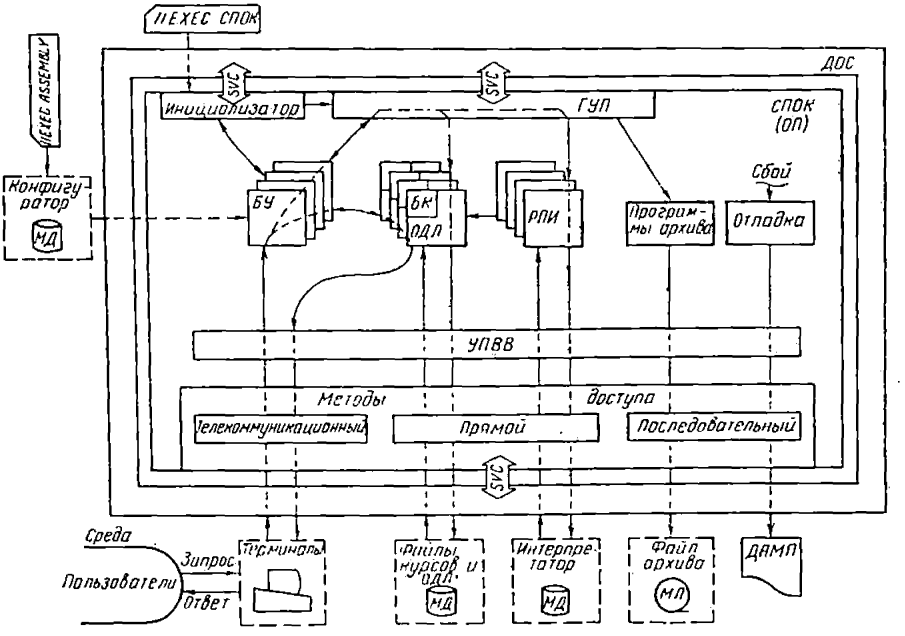



Рис. 5. Функционально-структурная схема СПОК

На этапе генерации по макрокоманде CANAL определяется столько БУ, сколько терминалов предполагается иметь в данной конфигурации СПОК, т. е. блок управления однозначно соответствует определенному терминалу (или линии). В понятие линии мы включаем совокупность технических устройств, необходимую для соединения терминала с процессором ЭВМ.

Для организации работы в режиме множественного доступа БУ должен содержать информацию об общем состоянии линии, операциях с диском и лентой, а также об операциях телеобработки.

В ОДЛ содержатся данные о категории пользователя, занимающего данную линию (терминал); данные о курсе; блок курса (БК) в 512 байт, с которыми работает в данный момент пользователь — автор или обучаемый.

В системе ОДЛ является связующим звеном между управляющей программой, интерпретатором и файлом курсов. При функционировании системы число ОДЛ равно числу пользователей, работающих со СПОК. Поэтому с целью экономии основной памяти используется алгоритм управления числом ОДЛ, размещающий в ОП небольшое их количество (задаваемое параметром LDAS макрокоманды CORE) и содержащий остальные ОДЛ на диске. По мере необходимости управляющая программа переводит требуемые ОДЛ с диска в ОП.

Рабочие поля интерпретатора, используемые управляющей программой, описаны выше.

Третий тип объектов, с которыми работает управляющая программа:

собственные программы или подсистемы СПОК, рассмотренные выше;

программы ДОС, включенные в СПОК либо на этапе генерации, либо при редактировании в режиме AUTOLINK.

К программам ДОС, включенным в СПОК, относятся программы телекоммуникационного, прямого и последовательного методов доступа (см. рис. 5), а также макрорасширения некоторых макрокоманд связи с супервизором (в частности, макрокоманды УСТАНОВИТЬ ВРЕМЯ — SETTIME, ВЫПОЛНИТЬ КАНАЛЬНУЮ ПРОГРАММУ — EXCP и др.). Макрорасширения всегда содержат команду обращения к супервизору SVC, с помощью которой запрашивается выполнение некоторой функции супервизором ДОС.

Указанные программы ДОС, программы архива и управляющая программа являются резидентными программами СПОК.

Управляющая программа СПОК состоит из двух основных частей:

главной управляющей программы (ГУП), организующей выполнение работ в основной памяти;

управляющей программы ввода-вывода (УПВВ), организующей выполнение работ во внешней памяти, на терминалах и АЦПУ.

Основной функцией ГУП, как управляющей программы пакета реального времени, реализующего режим разделения времени на многих пользователей, является просмотр очереди сообщений пользователей и обработка этих сообщений с приемлемым временем реакции.

Как уже упоминалось, сообщения пользователей размещаются в БУ. ГУП просматривает очередь БУ, определяя тот, в котором имеется признак конца введенного сообщения. При нахождении такого БУ возможны следующие четыре варианта обработки сообщений.

**Вариант 1.** Все программы, требуемые для обработки сообщения, находятся в основной памяти. В этом случае ГУП инициализирует выполнение операций, необходимых для обработки сообщения, ожидает конца обработки, обращается к УПВВ для выдачи

ответа СПОК на терминал пользователя и переходит к анализу следующего БУ в очереди.

**Вариант 2.** Для обработки сообщения необходимо обращение к файлу на диске или ленте, причем этот файл не занят другой линией. В этом случае ГУП обращается к УПВВ, инициализируя нужную операцию ввода-вывода и переходит к анализу следующего БУ в очереди, не ожидая конца обработки.

**Вариант 3** отличается от варианта 2 тем, что файл, к которому необходимо обратиться, занят другой линией. Поэтому операция ввода-вывода не инициализируется, а ГУП переходит к анализу следующего БУ в очереди.

**Вариант 4** отличается от варианта 2 тем, что при обращении к файлу возникают сбои. В этом случае ГУП, обращаясь к УПВВ, многократно пытается инициализировать операцию ввода-вывода. И если не удается запустить операцию нормально, осуществляет логическое отключение данной линии, удалив соответствующий БУ из очереди. Об отключении линии пользователю посылается сообщение.

При работе ГУП не ожидает завершения операций ввода-вывода. Поэтому другой важной функцией ГУП при работе СПОК в разделах F1 и F2 является выделение времени работы разделам низшего приоритета. Если, просмотрев очередь БУ, ГУП не находит готовых к обслуживанию данных блоков, она с помощью макрокоманды SETTIME выделяет время для работы разделов низшего приоритета. Это время регулируется диспетчером СПОК с помощью специальной директивы, без которой нельзя организовать параллельную работу в других разделах, так как пакет СПОК на основе высшего приоритета использовал бы все время процессора.

И наконец, ГУП следит за правильностью работы других подсистем СПОК. При возникновении ошибок, влияющих на правильность работы всей системы, ГУП предпринимает попытки их исправить. И если это не удается, она вызывает программу отладки, которая осуществляет выдачу дампа. При этом работа пакета СПОК останавливается.

Управляющая программа ввода-вывода организует в системе проведение операций ввода-вывода для терминалов, диска, ленты и АЦПУ. Поэтому структурно она состоит из трех соответствующих частей, каждая из которых подготавливает данные для операций ввода-вывода и затем обращается к супервизору ДОС для выполнения соответствующих канальных программ. Для проведения операций ввода-вывода УПВВ всегда получает управление от ГУП, а при осуществлении данной операции возвращает ей управление.

Структурные части УПВВ относятся соответственно к телекоммуникационному, прямому и последовательному методам доступа (см. рис. 5). Программы этих методов доступа включаются в состав резидентных программ СПОК на этапе генерации либо при редактировании.

### ЭКСПЛУАТАЦИОННАЯ ДОКУМЕНТАЦИЯ СПОК

Эксплуатационная документация СПОК содержит печатные материалы, необходимые для проведения этапов генерации, инициализации и собственной работы.

Комплект документации включает инструкции для:
оператора, осуществляющего генерацию и запуск СПОК;
диспетчера — администратора системы;
авторов диалоговых прикладных программ СПОК (курсов);
пользователей курсов (обучаемых);
консультантов (преподавателей).

В комплект документации входит учебное пособие по методике программирования курсов на ЯОК. Кроме того, в библиотеке системы постоянно находится и доступен всем пользователям справочный курс СПРАВ1 по входному языку СПОК.

### ФУНКЦИОНИРОВАНИЕ СПОК

Ниже дано краткое описание функционирования СПОК на этапах генерации, инициализации и выполнения собственной работы.

Генерация пакета выполняется оператором системы (или ДОС) с помощью макрокоманд и программ конфигуратора, как это было описано выше.

Этап инициализации начинается после появления во входном потоке для программы управления заданиями ДОС ЕС карты //EXEC СПОК. ДОС загружает в ОП программу-инициализатор, которая выполняет функции, описанные выше, затем загружает в ОП резидентные программы СПОК, открывает необходимые файлы и передает управление ГУП.

ГУП начинает просмотр очереди БУ, определяя в каких из них появились законченные сообщения пользователей. Сообщения пользователей помещаются в БУ с терминалов программами телекоммуникационного метода доступа под управлением УПВВ. При обнаружении законченного сообщения в БУ ГУП передает управление УПВВ для чтения в РПИ соответствующей подпрограммы интерпретатора с диска.

Вызванная подпрограмма (это может быть и цепочка подпрограмм) обрабатывает сообщения, определяя одновременно его принадлежность к языкам директив пользователей, операторам ЯОК или ответам обучаемого.

Если принятое сообщение распознано как директива, то ГУП инициализирует исполнение в системе функций, предписываемых данной директивой. В частности, при этом ГУП может обращаться к различным подсистемам СПОК (см. рис. 5). Об окончании исполнения директивы СПОК информирует пользователя, посылая на терминал сообщение ПЕЧАТАЙТЕ ДИРЕКТИВУ.

Если принятое сообщение относится к операторам ЯОК, оно преобразуется интерпретатором в ОДЛ к виду, удобному для по-

следующего выполнения, и пересылается в файл библиотеки курсов в место, выделенное соответствующему курсу. Об окончании выполнения этих процедур СПОК информирует автора выдачей на терминал номера следующего оператора курса до тех пор, пока автор не введет директиву END.

Наконец, анализ сообщения пользователя курса, не являющегося директивой, т. е. анализ ответа, проводится с помощью блока курса либо находящегося в БК (см. рис. 5), либо считываемого в эту область ОП из библиотеки курсов. Первое считывание блоков курса происходит при регистрации его пользователя, а в дальнейшем осуществляется автоматически по мере продвижения пользователя по курсу. Признаком окончания обработки ответа является появление на терминале нового текста — справочной или учебной информации, указания о дальнейших действиях, сообщения об ошибках в ответе и др.

Множественный доступ пользователей к системе реализуется ГУП на основе анализа БУ с использованием ОДЛ и РПИ. Напомним, что БУ в очереди столько, сколько имеется в данный момент активных терминалов пользователей.

Каждое сообщение пользователя обрабатывается ГУП в соответствии с вариантами, указанными выше, в режиме разделения времени центрального процессора на обработку сообщений. При этом ГУП не выделяет фиксированный квант времени на обработку сообщения, это время зависит от объема работ, которые согласно одному из вариантов, необходимо выполнить системе.

Применяемая в СПОК дисциплина циклического просмотра управляющей программой очереди БУ позволяет минимизировать время на подготовку к обработке сообщений, а следовательно, и суммарное время обработки всех сообщений пользователей системы.

Указанный способ обработки сообщений позволяет достаточно просто организовать доступ пользователей к различным прикладным программам, хранящимся в библиотеке курсов. Например, каждый пользователь может работать со своей программой либо несколько пользователей могут работать с одной программой.

Если в процессе интерпретации операторов курса возникает необходимость сохранить информацию в файле архива, ГУП передает управление программе архива.

При обнаружении неустранимых сбоев ГУП вызывает программу отладки и передает ей управление.

### О ВХОДНОМ ЯЗЫКЕ СПОК

Входной язык СПОК состоит из проблемно-ориентированного языка описания курсов и языков директив указанных выше четырех категорий пользователей СПОК.

Прототипами являются следующие языки: КОДИАЛ [14], COURSEWRITER III [15], TUTOR [6] и др.

При разработке входного языка СПОК учитывались системологические требования простоты взаимодействия СПОК с внешней средой, возможность расширения ЯОК специальными операторами,

необходимыми авторам прикладных программ, а также возможность «обучения» всей системы за счет средств ЯОК с целью придания ей требуемой функциональной ориентации.

### ЯЗЫК ОПИСАНИЯ КУРСОВ

С данным языком могут работать не только профессиональные программисты, но и авторы курсов, не имеющие подготовки в области языков программирования, например преподаватели средней и высшей школ и курсов повышения квалификации. Использование этого языка представляет авторам курсов достаточно простые средства управления обучением и позволяет в доступной и удобной форме записывать различные алгоритмы контроля ответов обучаемого.

В курсах обучения, которые могут быть описаны с помощью ЯОК, реализуется взаимодействие с пользователем, предполагающее:

выдачу обучаемому текста для изучения, заканчивающегося вопросом, инструкцией или заданием;

прием ответа обучаемого на поставленное задание, анализ правильности полученного ответа;

выдачу сообщения о результатах анализа ответа и принятие решения о дальнейшем ходе обучения или «беседы».

ЯОК не привязан ни к одной методике обучения или диалога, в связи с чем структура курса и использование той или иной методики целиком зависят от усмотрения автора.

Единицей ЯОК является оператор со следующим форматом:

$$\triangleright \text{<код—операции>}[\sqcup \text{текст }] \text{ Вв},$$

где $\triangleright$ — символ начала ввода;

Вв — символ конца ввода.

Квадратные скобки показывают, что в некоторых операторах текстовое поле может отсутствовать и подобный оператор состоит только из кода операции.

Функционально все операторы ЯОК можно разделить на следующие группы:

выдачи текстов (порций учебного материала, вопросов, сообщений о результатах работы прикладной программы и т. п.) на терминал;

приема и контроля сообщений пользователя курса, не являющихся директивами (в том числе ответов обучаемых);

сбора и обработки статистических данных о работе пользователя с курсом;

расширения ЯОК (оператор FN);

редактирования ответов обучаемого;

перехода.

Для изменения функций некоторых операторов в ЯОК существует набор модификаторов. Оператор с модификаторами имеет следующий формат:

$$\triangleright \text{<код—операции>} \ (m_1, \ldots, m_n) \ [\sqcup \text{текст}] \text{ Вв},$$

где $m_1, \ldots, m_n$ — любые допустимые в языке модификаторы.

ЯОК включает макросистему, позволяющую авторам описывать структуру некоторых фрагментов программы курса в виде макроопределений и обращаться к ним из любого места курса посредством макрокоманд с заданными параметрами.

Если в обучающем курсе содержатся фрагменты с одинаковой структурой, отличающиеся только содержимым текстовых полей операторов, то при программировании такого курса удобно использовать макросистему ЯОК.

Структура курса в общем виде показана на рис. 6. Каждый обучающий курс может быть разделен на несколько сегментов. На рис. 6 курс назван ПРИМЕР и сегменты обозначены соответственно ПРИМЕР-00 и ПРИМЕР-01. Разделение курса на сегменты позволяет нескольким авторам одновременно программировать и корректировать один и тот же курс.

```
ПРИМЕР-00/ИВАНОВ                12.05.77
МЕТ1
      1—0 . . . .
      1—1 . . . .
      1—2 . . . .
      1—3 . . . .
      2—0 . . . .
      2—1 . . . .
      2—2 . . . .
      . . . . . .
МЕТ2
      1—0 . . . .
      1—1 . . . .
      . . . .
      2—0 . . . .
      . . . .
      3—0 . . . .
      . . . . . .
      . . . .
      . . . . . .
МЕТn
      1—0 . . . .
      . . . .
END
MACROS
СЧЕТ
      1—0 . . . .
      1—1 . . . .
      . . . .
СЧЕТ1
      1—0 . . . .
      . . . .
      . . . .
      . . . .
MACEND
ПРИМЕР-01/ПЕТРОВ                12.05.77
МЕТКА1
      . . . . . .
      . . . . . .
МЕТКА
      . . . . . .
```

Рис. 6. Структура курса в общем виде

Внутри каждого сегмента программа курса может быть разделена на отдельные фрагменты с помощью меток. Число меток в сегменте произвольно. Метка представляет собой буквенно-цифровую последовательность от 1 до 6 символов, начинающуюся с буквы, и служит для обозначения места в программе, на которое может быть сделан переход при изменении последовательности выполнения курса.

При вводе программы курса все операторы автоматически пронумеровываются системой. Номер оператора состоит из двух чисел, между которыми стоит дефис. Первое число является номером оператора, начинающего группу вопроса, задания и т. п. внутри данной метки. Второе число — номер оператора внутри группы вопроса. В частности, группой вопроса может быть последовательность операторов, состоящая из оператора выдачи обучаемому текста для изучения и вопроса по этому тексту; операторов контроля ответа на поставленный вопрос; операторов выдачи сообщений о результатах контроля. Кроме того, в группу операторов вопроса могут входить операторы сбора статистики по данному вопросу и операторы перехода.

Следующий оператор выдачи текста для изучения или вопроса начинает новую группу вопроса.

При регистрации курса в библиотеке СПОК в поле курса автоматически проставляются четыре метки: имя курса (первая метка), метка END, отмечающая конец курса, метки MACROS и MACEND, отмечающие соответственно начало и конец поля макроопределений. Метки в поле макроопределений являются именами макроопределений, описанных автором. По этим меткам (именам) идет обращение из макрокоманд. Кроме того, при регистрации курса для каждого его пользователя отводится шесть специальных областей памяти: буферы, счетчики, переключатели, переключатели параметров, регистры возврата и счетчики времени.

*Буферы.* Шесть буферов (B0 — B5) по 100 байт каждый предназначены для хранения буквенно-цифровой информации в символьном виде. В буфере B0 всегда содержится последний ответ обучаемого. Буферы B1 — B5 автор курса может использовать для размещения в них текстов, которые могут затем быть посланы пользователю курса.

*Счетчики.* На каждого пользователя курса отводится 31 двухбайтный счетчик (C0 — C30), предназначенный для хранения числовой информации. В счетчике C0 всегда содержится время ответа пользователя курса на последний вопрос задания. Остальные счетчики могут использоваться для накопления других статистических данных о работе пользователя с курсом.

*Счетчики времени.* В счетчиках времени содержится следующая информация: в T0 — текущее время дня, в T1 — общее время работы обучаемого с курсом, в T2 — время с момента последнего подключения обучаемого к курсу. Автор может только пользоваться информацией, содержащейся в счетчиках времени. Изменять эту информацию он не может.

*Переключатели.* Общая емкость 32 переключателей (S0 — S31) — 4 байта. Каждый из переключателей находится в одном из двух состояний: включен (1) или выключен (0). Переключатели могут использоваться автором курса для того, чтобы зафиксировать результат выполнения или невыполнения какого-то условия. Впоследствии, согласно состоянию того или иного переключателя, может быть осуществлен переход на тот или иной фрагмент курса, т. е. переключатели дают возможность изменять порядок выполнения курса.

*Переключатели параметров* аналогичны переключателям. Но включение или выключение тех или иных переключателей параметров вызывает выполнение определенных действий в программе курса. Обозначаются P0 — P31.

*Регистры возврата.* Шесть регистров возврата (R0 — R5) по 6 байт каждый предназначены для хранения меток курса, на которые в процессе выполнения программы курса может осуществляться переход.

Кроме описанных специальных областей памяти, по желанию автора для каждого пользователя курса может быть отведено до 127 блоков вспомогательной памяти (1 блок=512 байт). Вспомогательная память может загружаться так же, как буферы или счетчики. Но работать с загруженной в эту память информацией можно только через буферы или счетчики.

**Пример программы на ЯОК.** На рис. 7 показан фрагмент обучающего курса по языку Фортран. При работе с приведенным фрагментом курса обучаемому будет выдан текст оператора вопроса (QU). После выдачи вопроса система переходит в состояние ожидания ответа. Следующий за оператором QU оператор ED ALL удаляет из введенного обучаемым ответа все символы пробела и новой строки, которые могут оказаться в ответе. Далее три оператора загрузки (LD) помещают нули в счетчики 3, 4 и 5, в которых по ходу работы обучаемого будет накапливаться информация о количестве соответственно правильных, неправильных и непредполагаемых ответов. Последний оператор LD загружает содержимое текстового поля в буфер B2.

Принятый ответ обучаемого сравнивается последовательно с содержимым каждого оператора контроля ответа (CA и WA). Первым стоит оператор правильного ответа (CA). Модификатор W в этом операторе указывает на то, что в текстовом поле оператора помещено несколько эталонов ответов и что ответ обучаемого должен последовательно сравниваться с каждым из них. При совпадении ответа с одним из эталонов оператора CA будут выполнены следующие за ним:

оператор FN CT, который вызовет стирание экрана;

оператор TY, выдающий на экран сообщение, находящееся в его текстовом поле;

оператор AD, по которому к содержимому C3 будет прибавлена единица.

После этого автоматически произойдет переход на оператор PR.

```
ПРАВ5
1—   0   QU      ПРАВИЛО 5.
1—   1           В ЯЗЫКЕ ФОРТРАН НЕЛЬЗЯ СМЕШИВАТЬ ВЕЛИЧИ-
1—   2           НЫ РАЗНЫХ ТИПОВ В ОДНОМ ВЫРАЖЕНИИ. ИСК-
1—   3           ЛЮЧЕНИЕ СОСТАВЛЯЕТ ОПЕРАЦИЯ ВОЗВЕДЕНИЯ
1—   4           В ЦЕЛУЮ СТЕПЕНЬ ВЕЩЕСТВЕННОГО ЧИСЛА. З А-
1—   5           Д А Н И Е: В ПРИВЕДЕННОМ НИЖЕ ВЫРАЖЕНИИ
1—   6           ДОПУЩЕНА ОШИБКА. НАЙДИТЕ ЕЕ И ВВЕДИТЕ ПРА-
1—   7           ВИЛЬНОЕ ВЫРАЖЕНИЕ
1—   8           A+2.5*B**3 — C/2   *0.37*B
1—   9   ED      AIL
1—  10   LD      0/C3
1—  11   LD      0/C4
1—  12   LD      0/C5
1—  13   LD      НА ЭТОТ ВОПРОС ВЫ ОТВЕТИЛИ С ПОПЫТОК/B2
1—  14   CA(W)   A+2.5*B**3 — C/2.0*0.37*B  A+2.5*B **3 — C/2.*
1—  15           0.37*B
1—  16   FN      CT
1—  17   TY      СОВЕРШЕННО ВЕРНО. ВЕЩЕСТВЕННУЮ ПЕРЕМЕН-
1—  18           НУЮ С НЕЛЬЗЯ ДЕЛИТЬ
1—  19           НА ЦЕЛУЮ КОНСТАНТУ 2.
1—  20   AD      1/C3
1—  21   CA(W)   A+2.5*B**3.0 — C/2.& &B*3. — C/2.& &B*3.0 — C/2.0&
1—  22   FN      CT
1—  23   TY      ВАШ ОТВЕТ, В ПРИНЦИПЕ, ВЕРЕН, НО ВЫ ВОЗВОДИ-
1—  24           ТЕ ПЕРЕМЕННУЮ В ВЕЩЕСТВЕННУЮ СТЕПЕНЬ.
1—  25           ЭТО ДЕЛАТЬ НЕ ОБЯЗАТЕЛЬНО, Т. К. ВЕЩЕСТВЕН-
1—  26           НУЮ ПЕРЕМЕННУЮ МОЖНО ВОЗВОДИТЬ В ЦЕЛУЮ
1—  27           СТЕПЕНЬ.
1—  28   AD      1/C3
1—  29   WA(W)   A+2.5*B**3. — C/2*0.37*B
1—  30           A+2.5*B **3.0 — C/2*0.37*B
1—  31   AD      1/C4
1—  32   TY      НЕВЕРНО. ВЫ СДЕЛАЛИ ВЕЩЕСТВЕННЫМ ПОКАЗА-
1—  33           ТЕЛЬ СТЕПЕНИ, В КОТОРУЮ ВОЗВОДИТСЯ ПЕРЕ-
1—  34           МЕННАЯ В, НО ВЕДЬ ВЕЩЕСТВЕННОЕ ЧИСЛО МО-
1—  35           ЖЕТ ВОЗВОДИТЬСЯ В ЦЕЛУЮ СТЕПЕНЬ. ПОДУМАЙ-
1—  36           ТЕ И ОТВЕТЬТЕ ЕЩЕ РАЗ
1—  37   UN      МНЕ НЕ ПОНЯТНО. ПОПЫТАЙТЕСЬ ЕЩЕ РАЗ ОТВЕ-
1—  38           ТИТЬ НА ВОПРОС.
1—  39   AD      1/C5
1—  40   UN      ВЫ ВТОРОЙ РАЗ ДАЕТЕ ОТВЕТ, КОТОРЫЙ Я НЕ ПО-
1—  41           НИМАЮ. ЕСЛИ НЕ МОЖЕТЕ ОТВЕТИТЬ НА ЭТОТ
1—  42           ВОПРОС, ЗАПРОСИТЕ ПОМОЩЬ
1—  43   AD      1/C5
2—   0   PR
2—   1   AD      C3/C4
2—   2   AD      C5/C4
2—   3   LD      C4/B2,29,2
2—   4   TY      B2
2—   5   BR      ПРАВ5/C4.CE.3
3—   0   QU(N)   НАЙДИТЕ ОШИБКУ В СЛЕДУЮЩЕМ ПРИМЕРЕ И
3—   1           ВВЕДИТЕ ПРАВИЛЬНЫЙ ОТВЕТ!
3—   2           2A/C**—2—COS(3A)
3—   3   LD      0/C3
3—   4   ED      ALL
3—   5   EP
3—   6   CA(W)   2.*A/C**2 — COS(3.*A)  2.0* A/C**2 — COS(3.*A)
3—   7           2.*A/C**2 — COS(3.0*A)  2.0* A/C**2 — COS(3.*A)
```

```
3—  8   FN.     CT
3—  9   AD      1/C3
3— 10   TY      ПРАВИЛЬНО.
3— 11   WA(W)   2*A/C*2 — COS(3*A)  2A/C*2 — COS(3*A)
3— 12           2*A/C*2 — COS(3A)
3— 13   AD      1/C3
3— 14   TY      НЕВЕРНО. ВСПОМНИТЕ ВСЕ ПРОЙДЕННЫЕ ПРАВИ-
3— 15           ЛА ЗАПИСИ ВЫРАЖЕНИЙ И ОТВЕТЬТЕ ЕЩЕ РАЗ.
3— 16   UN      ВЫ ОШИБЛИСЬ. ВНИМАТЕЛЬНО ПОСМОТРИТЕ НА
3— 17           ЗАДАНИЕ И ПОСТАРАЙТЕСЬ ОТВЕТИТЬ ПРАВИЛЬНО.
3— 18   AD      1/C3
3— 19   UN      ОПЯТЬ НЕВЕРНО. ЗАПРОСИТЕ ПОМОЩЬ, ЕСЛИ НЕ
3— 20           МОЖЕТЕ ОТВЕТИТЬ САМИ.
3— 21   AD      1/C3
4—  0   PR
4—  1   LD      C3/B2,29,2
4—  2   TY      B2
4—  3   BR      ПРАВ6/C3,LE,2
4—  4   TY      ВЫ СЛИШКОМ МНОГО ОШИБАЛИСЬ ПРИ ОТВЕТЕ
4—  5           НА ЭТОТ  ВОПРОС. ВАМ СЛЕДУЕТ ПОВТОРИТЬ ПЕР-
4—  6           ВЫЕ ПЯТЬ ПРАВИЛ ЗАПИСИ АРИФМЕТИЧЕСКИХ ВЫ-
4—  7           РАЖЕНИЙ НА ФОРТРАНЕ, ПРЕЖДЕ ЧЕМ ПЕРЕЙТИ К
4—  8           СЛЕДУЮЩЕЙ ПОРЦИИ.
4—  9   BR      ПОВТОР
```

Рис. 7. Фрагмент обучающего курса по языку Фортран

Если ответ не совпадает ни с одним из эталонов оператора CA, следующие за ним операторы FN, TY и AD игнорируются и выполняется оператор неправильного ответа (WA). При совпадении с одним из неправильных ответов выполняются следующие за WA операторы AD и TY вплоть до первого оператора UN и управление будет передано на точку возврата, т. е. на точку ожидания нового ответа обучаемого на поставленный вопрос.

Если ответ обучаемого не совпадает ни с одним из имеющихся эталонов ответа, выполняется оператор UN, т. е. посылается текст этого оператора, выполняется следующий за ним оператор AD и управление передается на ожидание очередного ответа. При следующем непредполагаемом ответе обучаемому посылается текст второго оператора и т. д. до тех пор, пока обучаемый не введет правильный ответ.

После ввода правильного ответа выполняются операторы сложения (AD C3/C4 и AD C5/C4) содержимого счетчиков 3, 4 и 5. Затем полученное число посылается в буфер 2, где хранится текст о количестве попыток, и оператором TY этот текст выдается обучаемому. Если количество попыток больше или равно 3, происходит переход на метку ПРАВ5, т. е. обучаемому предлагается повторить предыдущую порцию. В противном случае он переходит на следующий по порядку оператор QU.

Структура следующего фрагмента похожа на предыдущий фрагмент. Только с оператором QU используется модификатор N, указывающий на то, что данный оператор не может быть точкой возобновления работы при подключении. Дело в том, что при прохождении курса обучаемый может отключиться от системы в любой мо-

мент, когда система ожидает ввод с терминала. Информация о работе обучаемого автоматически запоминается и при следующем подключении к системе обучаемый начинает работу с того места, где он остановился прошлый раз, т. е. с последнего выданного вопроса. Если же в этой группе вопроса с оператором QU используется модификатор N, обучаемый в описанной выше ситуации возвратится не на этот оператор QU, а на предыдущий.

В конце фрагмента обучаемый перейдет на метку ПРАВ6, если он ответил на вопрос не более чем с двух попыток. В противном случае он отправляется на метку ПОВТОР, где ему придется повторить ранее пройденные правила. На рис. 8 показан пример работы обучаемого с описанным фрагментом курса. Сообщения обучаемого подчеркнуты.

ПРАВИЛО 5.
В ЯЗЫКЕ ФОРТРАН НЕЛЬЗЯ СМЕШИВАТЬ ВЕЛИЧИНЫ РАЗНЫХ ТИПОВ В ОДНОМ ВЫРАЖЕНИИ. ИСКЛЮЧЕНИЕ СОСТАВЛЯЕТ ОПЕРАЦИЯ ВОЗВЕДЕНИЯ В ЦЕЛУЮ СТЕПЕНЬ ВЕЩЕСТВЕННОГО ЧИСЛА.
ЗАДАНИЕ: В ПРИВЕДЕННОМ НИЖЕ ВЫРАЖЕНИИ ДОПУЩЕНА ОШИБКА. НАЙДИТЕ ЕЕ И ВВЕДИТЕ ПРАВИЛЬНОЕ ВЫРАЖЕНИЕ
A+2.5*B**3 — C/2 *0.37*B
A+2.5*B**3.0 — C/2*0.37*B
НЕВЕРНО. ВЫ СДЕЛАЛИ ВЕЩЕСТВЕННЫМ ПОКАЗАТЕЛЬ СТЕПЕНИ, В КОТОРУЮ ВОЗВОДИТСЯ ПЕРЕМЕННАЯ B, НО ВЕДЬ ВЕЩЕСТВЕННОЕ ЧИСЛО МОЖЕТ ВОЗВОДИТЬСЯ В ЦЕЛУЮ СТЕПЕНЬ. ПОДУМАЙТЕ И ОТВЕТЬТЕ ЕЩЕ РАЗ.
A+2.5*B**3 — C/2.0*0.37*B
СОВЕРШЕННО ВЕРНО. ВЕЩЕСТВЕННУЮ ПЕРЕМЕННУЮ С НЕЛЬЗЯ ДЕЛИТЬ НА ЦЕЛУЮ КОНСТАНТУ 2.
НАЙДИТЕ ОШИБКУ В СЛЕДУЮЩЕМ ПРИМЕРЕ И ВВЕДИТЕ ПРАВИЛЬНЫЙ ОТВЕТ:
2A/C**2 — COS(3A)
2*A/C*2— COS (3*A)
НЕВЕРНО. ВСПОМНИТЕ ВСЕ ПРОЙДЕННЫЕ ПРАВИЛА ЗАПИСИ ВЫРАЖЕНИЙ И ОТВЕТЬТЕ ЕЩЕ РАЗ.
2.0*A/C*2.0 — COS (3A)
ВЫ ОШИБЛИСЬ. ВНИМАТЕЛЬНО ПОСМОТРИТЕ НА ЗАДАНИЕ И ПОСТАРАЙТЕСЬ ОТВЕТИТЬ ПРАВИЛЬНО.
2.*A/C**2 — COS (3.*A)
ПРАВИЛЬНО.
ВЫ СЛИШКОМ МНОГО ОШИБАЛИСЬ ПРИ ОТВЕТЕ НА ЭТОТ ВОПРОС. ВАМ СЛЕДУЕТ ПОВТОРИТЬ ПЕРВЫЕ ПЯТЬ ПРАВИЛ ЗАПИСИ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ НА ФОРТРАНЕ, ПРЕЖДЕ ЧЕМ ПЕРЕЙТИ К СЛЕДУЮЩЕЙ ПОРЦИИ

Рис. 8. Пример работы обучаемого с описанным фрагментом курса

При работе программы курса с обучаемым номера операторов курса не проставляются.

### ЯЗЫКИ ДИРЕКТИВ ПОЛЬЗОВАТЕЛЕЙ СПОК

Ниже подробно рассмотрены языки директив всех типов пользователей, работающих с системой СПОК: автора, пользователя курса, преподавателя и диспетчера. Формат директивы следующий:

▷<название — директивы> [␣аргументы ] Вв

Пользователь может вводить директиву только после получения сообщения системы ПЕЧАТАЙТЕ ДИРЕКТИВУ.

**Сообщения об ошибках в директивах пользователей.** Диагностические сообщения об ошибках в директивах пользователей могут быть двух типов:

стандартные сообщения об ошибках, являющиеся общими для всех директив;

сообщения об ошибках в определенной директиве.

К стандартным сообщениям относятся, например:

ОШИБКА В ДИРЕКТИВЕ

КУРС xxxxxx — yy НЕДОСТУПЕН ИЛИ НЕ ЗАРЕГИСТРИРОВАН

ЛИНИЯ ОТКЛЮЧЕНА

ПОВТОРИТЕ, ПОЖАЛУЙСТА

ОШИБКА В ФОРМАТЕ и т. д.

Сообщения об ошибках в определенной директиве указывают на конкретные ошибки.

**Язык директив автора** представляет собой набор директив, с помощью которых автор может подключаться и отключаться от СПОК, вводить и корректировать курс, описанный в ЯОК, выводить на устройство отображения информации фрагменты уже введенного курса, переходить из режима автора в режим пользователя курса и др.

Работа автора со СПОК всегда начинается с подключения к системе посредством директивы ПОДКЛЮЧИТЬ. Пример начала работы автора с системой:

| | |
|---|---|
| Автор: | ▷ПОДКЛЮЧИТЬВв |
| Система: | ВВЕДИТЕ НОМЕР |
| Автор: | ▷А32Вв |
| Система: | 12/11/77 10 : 05 ЛИНИЯ 2<br>ВВЕДИТЕ ИМЯ КУРСА |
| Автор: | ▷ПРИМЕРВв |
| Система: | ВАШЕ ИМЯ <имя автора> ОСТАЛОСЬ ппп БЛОКОВ<br>ПЕЧАТАЙТЕ ДИРЕКТИВУ |

Отключиться от системы автор может директивой ОТКЛЮЧИТЬ в любой момент, когда система ждет ввода директивы. Для ввода и корректировки курса в языке директив автора имеются директивы:

ВСТАВИТЬ ПОСЛЕ — для ввода материала курса в определенное место;

УДАЛИТЬ — для удаления из программы курса одного или нескольких операторов;

ЗАМЕНИТЬ — для замены одного или группы операторов;

ПЕРЕМЕСТИТЬ — для перемещения одного оператора или фрагмента курса из одного места программы в другое.

В операторах ввода и корректировки программы курса для иден-

тификации операторов используются их номера, автоматически проставляемые системой.

Для проверки работоспособности программы курса автор с помощью директивы ПЕРЕЙТИ К переходит в режим пользователя курса и начинает работу с метки, указанной в директиве. Возвратиться в режим автора можно посредством директивы АВТОР.

По директиве ПЕЧАТАТЬ на терминал автора будет выведен фрагмент программы курса, границы которого определены в директиве.

**Язык директив пользователей курса.** Директивы пользователя курса служат для подключения к СПОК, выбора курса для изучения, запроса о помощи, передачи автору комментариев о курсе и отключении от СПОК. После отключения система запоминает информацию, обеспечивающую пользователю возможность продолжить изучение курса с того места, где он остановился.

Подключение к системе осуществляется директивой ПОДКЛЮЧИТЬ. Пример подключения обучаемого к системе:

| | |
|---|---|
| Обучаемый: | ▷ПОДКЛЮЧИТЬВв |
| Система: | ВВЕДИТЕ НОМЕР |
| Обучаемый: | ▷S10Вв |
| Система: | 10/05/77 13 : 36 ЛИНИЯ 3<br>ПЕЧАТАЙТЕ ИМЯ КУРСА |
| Обучаемый: | ▷ПРИМЕР Вв |
| Система: | ВАША ФАМИЛИЯ <фамилия обучаемого><br><текст курса> |

Отключиться от системы пользователь может директивой ОТКЛЮЧИТЬ в любой момент, когда система ожидает ввода с терминала.

Если обучаемый не может правильно ответить на вопрос, он может воспользоваться директивой ПОМОЩЬ, по которой ему будет выдан правильный ответ.

Иногда автор может не разрешить пользоваться директивой ПОМОЩЬ в некоторых фрагментах своего курса. Если обучаемый вводит ПОМОЩЬ в таком месте, ему выдается сообщение ПОМОЩЬ ОТСУТСТВУЕТ.

В некоторых курсах авторы могут дать пользователям возможность по их усмотрению изменять последовательность прохождения курса. В этих случаях пользователю дается список меток, на которые он может перейти, используя директиву ПЕРЕЙТИ К.

В любой момент, когда система ожидает ввода, пользователь может послать комментарий автору о курсе, с которым он работает, путем ввода двух символов «at», за которыми следует сообщение. Пример:

| | |
|---|---|
| Система: | ОТВЕТ ВЕРЕН. СЛЕДУЮЩИЙ ВОПРОС... |
| Обучаемый: | ▷«at» ДУМАЮ, ЧТО ДЛЯ ЭТОГО ВОПРОСА ВЕРЕН И ОТВЕТ 12Вв |
| Система: | БЛАГОДАРЮ ЗА КОММЕНТАРИЙ |

Комментарий не считается ответом и записывается на системное устройство наряду с полной информацией о номере пользова-

теля, имени курса и его участка с тем, чтобы впоследствии автор знал, по какой части курса сделан комментарий.

**Язык директив консультанта (преподавателя).** В функции консультанта входит помощь пользователям курсов в эксплуатации терминала и СПОК, а также по содержательной части диалогового курса.

Подключение консультанта к СПОК несколько отличается от подключения автора и пользователя курса. В данном случае используется код полномочия, задаваемый при генерации системы. Пример:

| | |
|---|---|
| Консультант: | ▷М <код полномочия> Вв |
| Система: | 3/07/77 11 : 00 ЛИНИЯ 4 |
| | ПЕЧАТАЙТЕ ДИРЕКТИВУ |

Отключение от системы аналогично отключению других пользователей СПОК.

Для получения сведений о работе пользователей курсов в распоряжении консультанта имеются директивы СОСТОЯНИЕ ОБУЧАЕМОГО и СОСТОЯНИЕ ЛИНИЙ. По первой из этих директив на терминал преподавателя в табличной форме выдаются сведения о месте в курсе, до которого дошел обучаемый, когда он начал и когда последний раз работал с курсом, общее время работы с курсом, номер, фамилия пользователя и др.

Введя директиву СОСТОЯНИЕ ЛИНИЙ, консультант получает информацию о пользователях, работающих с системой в данный момент. Эта информация выводится в виде таблицы с колонками: номер линии, имя курса, номер группы, номер пользователя, дата, время.

**Язык директив диспетчера.** Директивы, используемые диспетчером при работе со СПОК, можно условно разделить на три группы: относящиеся к работе пользователей курсов; связанные с работой авторов и директивы управления работой всей системы в целом.

Подключение диспетчера к СПОК и отключение от нее происходят аналогично подключению и отключению консультанта, только перед кодом полномочия в директиве ПОДКЛЮЧИТЬ используется префикс Х.

В распоряжении диспетчера имеется директива ПОСЛАТЬ, с помощью которой диспетчер осуществляет связь с пользователями, работающими в данный момент с системой. По этой директиве всем или отдельным пользователям посылается сообщение, вводимое диспетчером.

*Директивы, управляющие работой пользователей курсов*

*Директива РЕГИСТРИРОВАТЬ ОБУЧАЕМОГО* используется для регистрации одного или группы обучаемых в зарегистрированном ранее курсе. Аргументами этой директивы являются:

имя курса;

номер пользователя, перед которым ставится префикс S;

имя пользователя;
номер группы (все обучаемые, работающие с данным курсом, могут быть разделены на группы);
режим записи статистических данных о работе пользователя с курсом и др.

*Директива УДАЛИТЬ ОБУЧАЕМОГО* «вычеркивает» пользователя из числа допущенных к указанному курсу. Аргументы директивы:

имя курса;
один из параметров:

а) дата (удаляются все пользователи, которые последний раз работали в указанном месяце);
б) метка (удаляются все пользователи, которые дошли при работе с курсом до указанной метки);
в) номер (удаляется пользователь, зарегистрированный под указанным номером).

*Директива ИЗМЕНИТЬ ДАННЫЕ ОБУЧАЕМОГО* позволяет изменять следующие аргументы, определенные во время регистрации обучаемого: номер группы, режим записи статистических данных и др.

*Директива СОСТОЯНИЕ ОБУЧАЕМОГО* выполняется аналогично такой же директиве преподавателя.

*Директива СМОТРЕТЬ СПЕЦ. ОБЛАСТИ ПАМЯТИ* выводит на терминал диспетчера специальные области памяти курса указанного пользователя. Аргументы директивы:

имя курса;
номер пользователя;
один из символов: B, C, P, R или S, определяющий поле памяти, содержимое которого должно быть показано (буферы, счетчики, переключатели, переключатели параметров, регистры возврата).

*Директива ПИСАТЬ СПЕЦ. ОБЛАСТИ ПАМЯТИ* дает возможность изменять содержимое поля специальной области памяти указанного пользователя. Аргументы такие же, как в директиве СМОТРЕТЬ СПЕЦ. ОБЛАСТИ ПАМЯТИ ОБУЧАЕМОГО.

*Директивы, связанные с работой авторов курсов*

*Директива РЕГИСТРИРОВАТЬ КУРС* используется для регистрации курсов в библиотеке СПОК. Аргументами директивы являются:

имя курса;
номер автора, перед которым стоит префикс A;
имя автора;
идентификатор файла на диске, в котором размещается курс;
число блоков, резервируемых для материала курса (1 блок= 512 байтам);
коэффициент таблицы меток;
символы частичной обработки ответа;
максимальное число пользователей, которые будут работать с курсом;

число блоков вспомогательной памяти, отводимое для каждого пользователя;
номер группы.

*Перекомпоновка курса*

При корректировке курса автором, т. е. при удалении, замене, перемещении операторов, все удаленные из курса операторы отмечаются флажками и исключаются при работе с этим курсом, физически оставаясь на диске. Число обращений к диску увеличивается, время реакции системы растет. При перекомпоновке операторы на диске размещаются в порядке их выполнения, удаленные логически операторы удаляются физически, что оптимизирует работу курса и уменьшает поле памяти, занимаемое этим курсом.

Для выполнения перекомпоновки используются *директивы УБРАТЬ КУРС И ПОМЕСТИТЬ КУРС*. Первая из этих директив вызывает перезапись курса с диска на ленту с перекомпоновкой. По директиве ПОМЕСТИТЬ КУРС перекомпонованный курс помещается с ленты на диск.

*Директива РАСПЕЧАТАТЬ* предназначается для записи на ленту материала всего курса или отдельных фрагментов. С подготовленной ленты в разделе BG с помощью утилиты ДОС DITTO можно распечатать на АЦПУ листинг соответственно всего курса или его фрагментов.

*Директива КОПИРОВАТЬ* используется для копирования определенного в директиве фрагмента курса в заданное место того же курса или в другой курс. Границы перемещаемого фрагмента, названия курсов, из которого и в который идет копирование и точное место в курсе (номер оператора), куда помещается копия фрагмента, задаются аргументами директивы КОПИРОВАТЬ.

*Директива АВТОВВОД* позволяет вводить материал курса, отперфорированный предварительно на перфокартах. Процесс автоввода включает три этапа:

подготовку колоды перфокарт с материалом курса;
перезапись этого материала с карт на ленту;
ввод материала с ленты на диск.

*Директивы управления работой системы*

*Директива ЛИНИИ* позволяет получить в табличной форме краткие сведения о загрузке линий в данный момент. На экран выдается строка с номерами линий, подключенных к СПОК. Под номерами работающих в данный момент линий проставляется код пользователя (А — автор, S — обучаемый, М — преподаватель, Х — диспетчер), а под номерами неработающих линий — дефис.

*Директива ПЕРЕЧИСЛИТЬ* выдает на терминал в табличной форме список курсов в файле и определенные сведения об этих курсах, причем с помощью данной директивы получают сведения об отдельном курсе, всех курсах в указанном файле и всех курсах во всех файлах.

В таблице, выдаваемой на экран по директиве ПЕРЕЧИСЛИТЬ,

приводятся следующие данные о курсе: номера первого и последнего блоков, отведенных для курса в файле; общее число блоков, занимаемых курсом, включая материал курса, записи автора и обучаемого, таблицу меток и т. д.; число блоков вспомогательной памяти; число использованных и свободных блоков материала курса; число использованных и свободных записей обучаемого; число использованных и свободных блоков таблицы меток; имя автора.

*Директива ПОДСЧИТАТЬ* подсчитывает общее число пользователей СПОК, работающих в данный момент с системой, и распределение их по группам, т. е. число авторов, обучаемых и т. д.

*Директива ВРЕМЯ ОТВЕТА.* По этой директиве на терминал диспетчера выводится информация о времени реакции системы при работе с пользователями курсов, причем выдается как среднее время реакции системы, так и спектр распределения времени реакции в процентном отношении.

*Директива ВРЕМЯ РАЗДЕЛА BG* позволяет диспетчеру изменять время, отводимое работам в разделах низшего приоритета (F2 и BG).

## Использование СПОК и перспективы развития системы

В настоящее время СПОК проходит опытную эксплуатацию в Институте кибернетики АН УССР, НИИ проблем высшей школы, Московском учебном центре АСУ, Казанском учебном центре СНПО «Алгоритм», Одесском политехническом институте и ряде других организаций.

Этими организациями разрабатываются диалоговые курсы, реализующие перечисленные выше типы методик автоматизированного обучения и обслуживания. В частности, разрабатываются курсы обучения операторов ОС ЕС, курсы обучения языкам программирования Кобол, Фортран и др.

Как показывает опыт эксплуатации, СПОК позволяет сократить время программирования и отладки диалоговых курсов в 5—10 раз по сравнению с программированием на языке ПЛ/1. Внедрение одного экземпляра СПОК обеспечивает экономический эффект порядка 50 тыс. руб. на одного автора диалоговых курсов в год.

В течение 1978—1979 гг. предполагается перевести СПОК на ОС ЕС, проверить работоспособность системы в режимах теледоступа. Предполагается также расширить ЯОК за счет включения функций, обеспечивающих контроль ответов обучаемых с помощью грамматик и возможность обращения к базам данных учебного назначения.

## ЛИТЕРАТУРА

1. Глушков В. М. и др. Человек и вычислительная техника. Киев, Наукова думка, 1971.

2. Довгялло А. М., Ющенко Е. Л. Обучение и диалоговое программирование с помощью ЭВМ. — УСиМ, 1974, № 1.

3. Довгялло А. М., Стогний А. А. Диалог человека и ЭВМ. М., Знание, 1975.

4. Tennyson R. D. Application of Computers in Education, «Andiovisual Instruction», May, 1974.
5. Довгялло А. М. и др. Обучение с использованием вычислительных машин: современное состояние и перспективы. — УСиМ, 1978, № 2.
6. Control Data PLATO, Seminar for Ministry of Higher and Specialized Secondary Education of USSR, Moscow, March — April, 1976.
7. Система программирования и ведения обучающих курсов (СПОК), «Прикладные программы (программы для ЕС ЭВМ)», 3—4 (11—12). М., ВИМИ, 1976.
8. Алексеенко Е. А. и др. СПОК — система программирования и поддержания обслуживающих и обучающих курсов. — УСиМ, 1978, № 2.
9. Дружинин В. В., Конторов Д. С. Проблемы системотологии. М., Советское радио, 1976.
10. Мартин Дж. Системный анализ передачи данных. Т. I. М., Мир, 1975.
11. Беспалько В. П. Программированное обучение (дидактические основы). М., Высшая школа, 1970.
12. Рынгач В. Д. и др. Обучающе-программирующая система ДИПРОФОР и некоторые результаты ее экспериментального исследования. — УСиМ, 1975, № 6.
13. Фатеев А. Е. и др. Прикладные программы в системе математического обеспечения ЕС ЭВМ. М., Статистика, 1976.
14. Белов В. Н., Довгялло А. М. Принципы организации и результаты экспериментального опробования пакета подпрограмм, ориентированных на изготовление диалоговых и обучающих программ. — УСиМ, 1978, № 1.
15. COURSEWRITER III, version 3, Anthor's Guide, SH20—1009, IBM Program Product, 1973.

*Ф. Л. ФРИДЛЕНДЕР, В. И. ГОРДОНОВА*

## ПОДСХЕМА ДЛЯ ЯЗЫКА ПЛ/1 В СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ СЕТЕВОЙ СТРУКТУРЫ

В настоящее время в ВГПТИ ЦСУ СССР разрабатывается система управления базами данных (СУБД), основанная на предложениях *DBTG CODASYL*, которые сформулированы в отчетах и рабочих записках 1971—1976 гг. [1, 2]. Концепция схемы и подсхемы, реализующая принцип интегрированного хранения и дифференцированного использования данных, является одной из важнейших в этих предложениях.

В предложениях *DBTG CODASYL* основное внимание уделено спецификациям схемы; подсхема и язык манипулирования данными (ЯМД) разработаны менее тщательно, поэтому они требуют более подробного рассмотрения при реализации СУБД. В отчетах *CODASYL* включающим языком предполагался Кобол.

В данной статье рассматривается подсхема, предназначенная для включающего языка ПЛ/1. В разрабатываемой системе подсхема не является частью прикладной программы. Подсхема, написанная на языке определения данных подсхемы (ЯОД для ПЛ/1), переводится во внутреннее представление компилятором подсхемы, являющимся одним из компонентов СУБД. При выполнении прикладной программы, взаимодействующей с базой данных, используется внутреннее представление подсхемы и требуемой части схемы. Отделение подсхемы от прикладной программы позволяет приме-

нять для компиляции прикладной программы стандартный компилятор с языка ПЛ/1. Подсхему могут использовать одна или несколько прикладных программ.

Подсхема, структура которой совпадает с предлагаемой в работе [1], состоит из пяти секций: идентификации, псевдонимов, области, набора и записи.

В секции идентификации определяется имя подсхемы, указывается имя схемы, с которой ассоциирована подсхема, могут быть определены замки управления доступом для использования подсхемы и операций над подсхемой, может быть задан ключ к замку управления доступом для компиляции подсхемы.

Секция псевдонимов предназначена для переименования тех или иных структур базы данных, определенных в подсхеме. Псевдонимы используются в секциях области, набора и записи подсхемы и в операторах языка манипулирования данными вместо соответствующих имен, определенных в схеме. Переименование может потребоваться в тех случаях, если в прикладной программе, использующей подсхему, приняты какие-то специальные соглашения об использовании имен. Секция псевдонимов необязательна, она не включается в подсхему, если переименование не требуется.

Секция области задает области базы данных, определенные в подсхеме. Могут быть перечислены имена этих областей или указано, что в подсхеме определены все области базы данных. Никакие характеристики областей не переопределяются в подсхеме. Прикладная программа не имеет доступа к экземплярам записей, расположенным в областях, которые не определены в подсхеме.

Секция набора задает наборы базы данных, определенные в подсхеме. Могут быть перечислены имена наборов или указано, что в подсхеме определены все наборы. В набор подсхемы могут быть включены не все записи-члены, определенные в схеме. Прикладной программе, использующей подсхему, набор представляется состоящим только из записей участников, определенных в подсхеме.

Если в схеме для записи-члена набора выражение выбора экземпляра набора (*SET SELECTION*) специфицирует выбор текущего экземпляра набора, то в подсхеме выражение выбора может быть переопределено. В других случаях переопределение не допускается, так как оно может нарушить структурирование базы данных, заданное в схеме. В работе [2] ограничение на переопределение выражения выбора экземпляра набора не дается.

Никакие характеристики набора, кроме описанных, не переопределяются в подсхеме. Секция набора необязательна. В тех случаях, если она не включена в подсхему, никакие наборы в подсхеме не определены.

Секции идентификации, псевдонимов, области и набора в подсхеме не зависят от включающего языка. На секцию записи включающий язык оказывает существенное влияние: структура записи в подсхеме должна соответствовать структурам, допустимым во включающем языке, так как описание записи в подсхеме порождает определение некоторой структуры (или структур) в программе на

включающем языке. Эти структуры называются ассоциированными с записью подсхемы.

В работах [1], [2] декларировалось, что запись подсхемы должна составляться из элементов одной или нескольких записей схемы. Однако для этих целей никакой аппарат не был предложен. Спецификации языка подсхемы для Кобола в работе [2] рассчитаны на то, что запись подсхемы формируется из элементов одноименной записи схемы[1]. Правила переопределения структуры записи в подсхеме не приведены.

В разрабатываемой системе возможность формирования записи подсхемы из элементов различных записей схемы реализуется описанным ниже способом.

Каждой записи, определенной в подсхеме, соответствует одноименная запись схемы — корневая запись по отношению к записи подсхемы.

Запись подсхемы может содержать элементы данных из корневой записи и записей схемы, которые могут быть связаны с корневой записью цепочкой следующей структуры. Пусть корневая запись определена в схеме членом некоторого набора, его владелец — членом другого набора и т. д. Последовательность записей — владельцев этих наборов образует искомую цепочку записей, элементы ко-

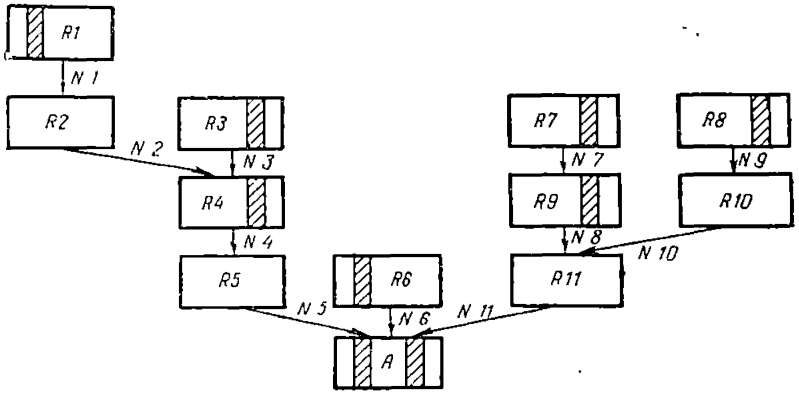



Рис. 1. Формирование записи подсхемы

торых могут быть включены в запись подсхемы (рис. 1). В соответствии с тем, как принято в ЯОД схемы, для одной корневой записи может существовать несколько таких цепочек. На рис. 1 прямоугольник представляет тип записи, стрелка — отношение типа записи-владельца к типу записи-члена набора. Имя записи указывается

[1] Здесь и далее считаем для удобства изложения, что переименование в подсхеме не осуществляется. Иначе с именами схемы следовало бы сопоставлять не имена, определенные в секции псевдонимов, а имена схемы, которым эти псевдонимы соответствуют.

внутри прямоугольника, имя набора — рядом со стрелкой. Прямоугольник А изображает корневую запись, заштрихованные участки прямоугольников — элементы данных, включаемые в запись подсхемы А. На рис. 1 показаны только те цепочки, которые участвуют в формировании записи подсхемы.

Экземпляр записи подсхемы порождается экземпляром корневой записи, так как экземпляр записи не может быть фактическим членом более чем одного экземпляра набора данного типа. Таким образом, структура на уровне типов (см. рис. 1) порождает аналогичную структуру на уровне экземпляров с тем только отличием, что некоторые цепочки, идущие от корневой записи, могут обрываться. Пусть, например, запись *R5*-владелец экземпляра набора *N5*, содержащего выбранный экземпляр корневой записи А, не является фактическим членом набора *N*4. Тогда значения элементов данных в соответствующем экземпляре записи подсхемы определяются отношениями, показанными на рис. 2. В отличие от рис. 1 здесь прямоугольник изображает экземпляр записи, а стрелка — отношение записи-владельца к записи-члену в экземпляре набора. Элементы данных записи подсхемы, формируемые из элементов данных записей *R*1, *R*3, *R*4, в рассматриваемом экземпляре записи подсхемы имеют значения ПУСТО.

Таким образом, если какая-либо запись в цепочке, определяющей запись подсхемы, не является фактическим членом соответствующего набора, то элементы данных, формируемые из элементов данных записи-владельца этого набора и наборов, расположенных «выше» в структуре записи, имеют значения ПУСТО.

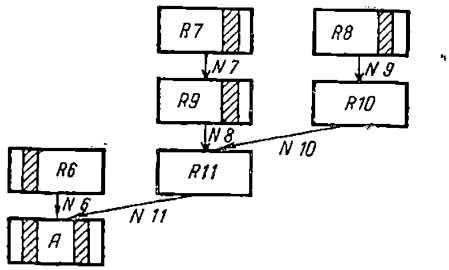



Рис. 2. Формирование экземпляра записи подсхемы

При запоминании записи в базе данных, поиске и модификации записи используются только те элементы данных записи подсхемы, которые формируются из данных корневой записи. При чтении записи в прикладную программу пересылаются из базы данных значения всех элементов данных, определенных в схеме и участвующих в формировании записи подсхемы согласно ее описанию.

Для удобства использования в прикладной программе структур, ассоциированных с записями подсхемы, в этих записях можно определить переменные, которым не соответствуют никакие имена данных базы данных. Эти переменные могут определять имена агрегатов данных, не предусмотренных в схеме, или именовать элементы данных, не являющиеся данными базы данных. Значения таких элементов не используются и не формируются при выполнении операторов ЯМД, за исключением переменных, которые содержат длины определенных в схеме элементов данных переменной длины.

При предтрансляции прикладной программы, т. е. переводе ее с расширенного языка, содержащего операторы ЯМД, на язык ПЛ/1, в программу включаются определения структур, ассоциированных с записями подсхемы. В этих структурах объявляются все переменные записей подсхемы, и прикладная программа может использовать эти переменные, как любые переменные включающего языка, независимо от того, соответствуют ли они каким-либо данным базы данных или нет.

Структура записи подсхемы аналогична структуре в языке ПЛ/1: запись может содержать скалярные переменные, массивы и другие структуры, допускаются массивы структур.

Для описания структуры используется предложение DATA-NAME, имеющее следующий формат:

номер — уровня имя данных — 1 [**FROM** имя — данных — 2

**USING** SETимя — набора — 1 [, имя — набора — 2]...]

$$\left[\left(\left\{\begin{array}{l}\text{целое} - 1\\ \text{идентификатор} - 1\end{array}\right\}\left[,\left\{\begin{array}{l}\text{целое} - 2\\ \text{идентификатор} - 2\end{array}\right\}\right]\ \ldots\right)\right]$$

Здесь использованы те же обозначения, что в работе [2]:

прописные латинские буквы — зарезервированные слова ЯОД для ПЛ/1;

строчные русские буквы — названия метапеременных, которые при написании конкретного предложения должны быть заменены соответствующими значениями;

обязательные зарезервированные слова подчеркнуты;

квадратные скобки означают, что заключенная в них конструкция может быть опущена;

фигурные скобки означают, что выбирается одна из заключенных в эти скобки конструкций;

многоточие показывает, что предшествующая конструкция может повторяться;

круглые скобки и запятая используются в ЯОД для ПЛ/1 как знаки пунктуации.

Номер уровня в предложении *DATA-NAME* имеет тот же смысл, что в языке ПЛ/1 — определяет номер уровня структуры и задается целым числом.

Имя-данных-1 есть имя переменной в записи подсхемы. Предполагается, что имена переменных внутри одной записи различны. Когда фрагмент *FROM*, т. е. заключенная в квадратные скобки конструкция, начинающаяся словом *FROM*, опущен, предложение *DATA-NAME* специфицирует включение в запись подсхемы данных из корневой записи, имеющих указанное имя, или определяет имя, не ссылающееся на имя данных базы данных, если имя-данных-1 не определено в корневой записи. Предложение *DATA-NAME* с фрагментом *FROM* специфицирует включение в запись подсхемы данных из записи схемы, не корневой по отношению к записи подсхемы. В этом случае перечисленные имена наборов задают описанную выше цепочку наборов, имя-данных-2 — имя данных в записи-владельце последнего набора. Эти данные включаются в запись

подсхемы под именем имя-данных-1. Переименование может оказаться необходимым во избежание совпадения имен в записи подсхемы.

Списки имен наборов в предложениях *DATA-NAME*, определяющих данные одной записи подсхемы, могут совпадать или один список может составлять часть другого. Целые и идентификаторы в предложении *DATA-NAME* задают верхние границы индексов массивов в записи подсхемы. Граница может быть постоянной, тогда она задается целым числом. Для классов памяти *AUTOMATIC* и *CONTROLLED* в языке ПЛ/1 допустимы массивы с переменной верхней границей. В этом случае верхняя граница индекса массива в экземпляре записи определяется значением элемента данных, на который ссылается соответствующий идентификатор в предложении *DATA-NAME*. Массив с переменной границей и переменная, задающая его границу, должны быть определены в одной записи подсхемы.

В соответствии с принятым в языке ПЛ/1 в ЯОД подсхемы для ПЛ/1 используется принцип определения по умолчанию. Чтобы включить в запись подсхемы определенный в схеме агрегат данных без какого-либо переопределения его характеристик, достаточно написать предложение *DATA-NAME* только для имени агрегата.

Формирование записи подсхемы из элементов данных, определенных в схеме, и переменных, не ссылающихся на данные базы данных, не должно нарушать некоторые отношения подчиненности, определенные в схеме.

Прежде чем перейти к формулировке правил переопределения структуры записи в подсхеме, введем некоторые понятия. Если имя данных в записи подсхемы ссылается на имя данных, определенное в схеме, будем называть второе имя прообразом первого. Имена данных в подсхеме, не ссылающиеся на данные базы данных, не имеют прообразов.

Если имена данных определяют соответственно группу и ее компонент (т. е. определение второго имени следует за определением первого, имеет больший номер уровня, чем у первого, и между ними нет определения данных с номером уровня, не превосходящим номер уровня первого), будем говорить, что первое имя порождает второе. Это определение справедливо и для схемы, и для подсхемы.

Имя-1 непосредственно порождает имя-2, если оно не порождает никакое имя-3, порождающее имя-2.

Возможность определения в подсхеме имени данных по умолчанию можно описать следующим образом: если в подсхеме явно (т. е. с помощью предложения *DATA-NAME*) определено имя, не порождающее других имен, а его прообраз порождает другие имена, то в подсхеме определены по умолчанию все эти имена схемы, и структура группы в подсхеме совпадает со структурой группы, имя которой есть прообраз имени в подсхеме.

Введем понятие прообраза идентификатора подсхемы (в соответствии с [1] под идентификатором понимается имя данных, уточненное до однозначности в пределах записи указанием в схеме но-

меров повторения всех содержащих эти данные повторяющихся групп, в подсхеме — индексов массивов, содержащих эти данные).

Для идентификатора без индексов понятие прообраза совпадает с понятием прообраза имени данных. Для идентификатора массива с заданным числом индексов прообраз определяется так. Рассматривается последовательность имен всех повторяющихся групп в схеме, порождающих прообраз имени массива и упорядоченных по убыванию номеров уровня. Прообраз имени массива включается в эту последовательность в том случае, если именует повторяющуюся группу или вектор. Прообразом идентификатора назовем имя, порядковый номер которого в этой последовательности равен числу измерений массива.

Для пояснения сказанного приведем пример. Пусть описание структуры записи в схеме имеет вид, приведенный ниже:

```
2 A ; OCCURS 5 TIMES ;
3 C ; OCCURS 2 TIMES ;
4 R ;
4 P ;
3 D ; OCCURS 3 TIMES ;
4 E ;
5 F ; OCCURS 7 TIMES ;
6 M ;
6 N ; OCCURS L TIMES ;
5 T ;
2 L ;
```

В данном описании атрибут *OCCURS* характеризует повторяющуюся группу ($A$, $C$, $D$, $F$) или вектор ($N$).

Пусть описание структуры одноименной записи в подсхеме имеет вид:

```
2 A  (5) ;
3 F  (2, 7) ;
4 M ;
4 N  (2) ;
4 K ;
```

Тогда имя $K$ не имеет прообраза, прообразы остальных имен определяются по совпадению имен в схеме и подсхеме. Имя $A$ в подсхеме порождает имена $F$, $M$, $N$, $K$ и непосредственно порождает только имя $F$. Имена $M$, $N$, $K$ не порождают никаких имен. Прообразом идентификатора $F$ (2, 7) является имя $D$, прообразом идентификатора $N$ (2) — имя $N$.

Если описание структуры записи подсхемы имеет вид:

```
2 M (5, 3, 7) ,
```

то прообразом идентификатора $M$ (5, 3, 7) является имя $A$.

При определении структуры записи в подсхеме должны соблюдаться следующие правила.

Имя в подсхеме, имеющее прообраз, не может порождать другое имя, если его прообраз не порождает никакое имя, и не может порождать другое имя с прообразом, если прообраз первого имени не порождает прообраз второго.

Имя в подсхеме должно быть индексированным, если его прообраз определен как повторяющаяся группа.

Если для индексированного имени в подсхеме $K$-й индекс справа задан идентификатором, имя индекса должно иметь прообразом коэффициент повторения прообраза идентификатора массива размерности $K$ с указанным индексированным именем. Имя индекса должно быть определено в подсхеме (явно или по умолчанию), причем индексированное имя и имя индекса должны порождаться разными именами структур уровня 1.

Для имени в подсхеме, имеющего прообраз, суммарное число индексов в самом имени и индексированных именах порождающих его массивов должно быть равным суммарному числу повторяющихся групп и векторов, порождающих прообраз имени массива или совпадающих с ним.

Если индексированное имя в подсхеме имеет прообраз и $K$-й индекс справа задан целым числом и если при этом повторяющаяся группа-прообраз идентификатора массива размерности $K$ с указанным индексированным именем имеет фиксированную длину, то значение индекса не должно превосходить коэффициент повторения группы.

Имя массива в подсхеме должно иметь прообраз, если оно порождает хотя бы одно имя, имеющее прообраз.

Имя в подсхеме не может непосредственно порождать имена, имеющие прообразы, если в схеме определены разные повторяющиеся группы, непосредственно порождающие прообразы соответствующих идентификаторов.

Имя данных, заданное в подсхеме, должно быть уникальным среди имен данных записи подсхемы, определенных явно или по умолчанию.

Примеры правильного переопределения структуры записи в подсхеме приведены ниже:

Пример 1

```
2 A (5) ;
3 B ;
4 C (2) ;
4 M (3, 7) ;
```

Пример 2

```
1 L ;
1 D (5, 3) ;
2 N (7, L) ;
```

Примеры неправильного переопределения структуры записи в подсхеме:

Пример 1

```
2 F (3, 1, 3) ;
3 M ;
4 K ;
```

Пример 2

```
2 F (3, 1, 3) ;
3 M ;
3 E ;
```

Пример 3

```
2 F (3, 1, 3) ;
3 N ;
```

Пример 4

```
2 F (3, K, 3) ;
2 K ;
```

Пример 5

```
2 A (3) ;
2 F (7) ;
```

Пример 6

```
2 A (5) ;
3 S (5) ;
4 C (2) ;
4 D (3) ;
```

Пример 7

```
2 A (4) ;
3 S ;
4 P ;
4 E ;
```

Пример 8

```
2 M (3, 4, 7) ;
```

Перейдем к описанию секции записи подсхемы. Секция записи задает записи, определенные в подсхеме. Имена записей могут быть перечислены явно или указано, что в подсхеме определены все записи.

В статье записи подсхемы может быть указан список областей, в которых прикладная программа, использующая подсхему, может размещать записи описываемого этой статьей типа при их запоминании в базе данных. В этом списке не должны содержаться имена областей, не определенных в подсхеме или не предназначенных для размещения соответствующей корневой записи согласно описанию в схеме. Заметим, что для чтения или модификации записей прикладная программа имеет доступ ко всем экземплярам записей, которые расположены в областях, определенных в подсхеме.

При описании записи подсхемы с помощью атрибутов *ALIGNED* или *UNALIGNED* могут быть заданы требования к выравниванию для структур, ассоциированных с записью подсхемы. Смысл этих атрибутов тот же, что в описании языка ПЛ/1 для ОС/ЕС.

Может быть указано, что определение структур, ассоциированных с записью подсхемы, не требуется. Заметим, что записи подсхемы в зависимости от ее описания может соответствовать одна или более структура уровня 1 в прикладной программе. Это оказывается необходимым для работы с массивами переменной размерности, так как в языке ПЛ/1 массив переменной размерности и переменная, определяющая его размерность, не могут являться компонентами одной и той же структуры.

Структура записи подсхемы задается последовательностью предложений *DATA-NAME*, как было указано выше.

Представление и область допустимых значений элемента данных задаются предложениями *TYPE* и *PICTURE*, аналогичными одноименным атрибутам в языке ПЛ/1.

Для элемента данных, имя которого имеет прообраз, предложения *TYPE* и *PICTURE* могут быть опущены, тогда задаваемая ими информация определяется из схемы. Если же предложение *TYPE* или *PICTURE* указано, это специфицирует необходимость преобразования данных при их пересылке из базы данных в прикладную программу или наоборот. Правила преобразования являются уточнением предложенных в [1], учитывающим специфику включающего языка. Стандартное преобразование может быть заменено специальным, выполняемым процедурой базы данных. Имя такой процедуры указывается при описании элемента данных в подсхеме.

В разрабатываемой системе не предусмотрено переопределение в подсхеме замков управления доступом, определенных в схеме, так как это может нарушить стратегию управления доступом, предусмотренную при создании базы данных.

## ЛИТЕРАТУРА

1. Data Base Task Group CODASYL, April, 71, Report.
2. CODASYL minutes of meeting, April, 1975.

*Г. К. СТОЛЯРОВ*

# ЯЗЫКИ И ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ СИСТЕМЫ АСПИД-3

Пакет прикладных программ (ППП) АСПИД-3 (автоматизированная система поиска информации по дескрипторам) предназначен для организации документальной информационно-поисковой системы дескрипторного типа с нормированной лексикой и поисковым файлом в виде инвертированных списков.

АСПИД-3 работает на ЭВМ серии ЕС под управлением операционной системы ДОС и обеспечивает справочно-информационное обслуживание в режиме избирательного распространения информации о новых поступлениях в соответствии с профилями интересов абонентов и в режиме ретроспективного поиска по всему файлу документов.

В зависимости от назначения и общей организации применения система позволяет генерировать различные варианты программы поиска, выбирать различные режимы функционирования программ, различные организацию, форматы и носители файлов, накапливать в архиве постоянные термы, запросы и серии запросов, хранить и выдавать в ответ на запросы полные поисковые образы и тексты документов либо только требуемые части текстов, номера или адреса документов, хранимых вне системы АСПИД-3 в других контурах обработки.

Информационно-вычислительные центры предприятий, организаций, библиотек и архивов могут использовать АСПИД-3 как автономно, так и в качестве подсистемы, с возможностью обращения к программе поиска из программ на Ассемблере и языках программирования высокого уровня (ПЛ/1, Кобол, Фортран). Это позволяет объединять в различных вариантах систему АСПИД-3 с программными средствами пользователя и осуществлять пред- и постобработку запросов и ответов системы.

В состав пакета прикладных программ АСПИД-3 входят 3 языка и 10 программ:

ЯРТ — язык работы с тезаурусом;
ЯРД — язык работы с документами;
ЯЗП — язык запросов на поиск;
А — программа создания (режим $A_1$) и обновления (режим $A_2$) тезауруса;
Х — программа распечатки тезауруса;
В — программа создания (режим $B_1$) и обновления (режим $B_2$) файла документов;
ВЗ — программа слияния файлов документов;
У — программа распечатки файла документов;
С1 — программа создания инвертированного поискового файла;

C2 — программа обновления поискового файла;
C3 — программа слияния поисковых файлов;
Z — программа распечатки частотных словарей дескрипторов;
D — программа поиска.

Кроме объектных модулей программ АСПИД-3, объектных модулей логической системы управления вводом-выводом (СУВВ) и управляющих операторов для редактирования программных компонентов в состав дистрибутивной ленты системы включены справочник для пользователей и учебный пример, демонстрирующий работу системы.

В применении системы АСПИД-3 можно выделить следующие основные этапы:

подготовку пользователя и системы;
анализ предметной области и создание тезауруса;
индексацию документов и создание файла документов;
создание инвертированного поискового файла;
подготовку запросов;
ввод запросов, поиск и выдачу ответов;
обработку ответов;
обновление и поддержку системных файлов;
поддержку системы и генерацию новых вариантов системы (реорганизация системных программ и файлов, разработка программ пред- и постобработки, накопление архива постоянных запросов).

Часть этих работ выполняется на языках и программами АСПИД-3 или с помощью этих средств. Часть работы выполняют системные специалисты-индексаторы, знакомые с предметной областью применения, и программисты. Для выполнения этих работ могут использоваться дополнительные программные средства пользователя, учитывающие системные требования и форматы.

В нижеследующих описаниях языков элементы языка, образующие операции, состоят из последовательностей прописных (заглавных) букв и специальных литер разделителей-ограничителей.

Элементы метаязыка, служащие для описания языка, но сами не являющиеся элементами языка, включают расширенные нотации Бэкуса — Наура.

Факультативные фрагменты заключаются в прямые скобки, альтернативные — в фигурные и разделяются наклонной чертой. Фрагменты в скобках, за которыми следуют многоточия, могут повторяться.

На схемах системных процедур, кроме общепринятых, использованы кружки для обозначения файлов безотносительно к их носителям, точки в кружках — для указания возможности создания индекса к соответствующим файлам, прямоугольники — для обозначения программ системы АСПИД-3, заштрихованные прямоугольники — для обозначения программных средств пользователя.

**Подготовка пользователя и системы** включает изучение описания системы, установку системы у пользователя (распечатка справочника, сброс программ пакета с дистрибутивной ленты и генерация требуемых вариантов программ) и упражнения с демонстрационным примером, входящим в состав дистрибутивной ленты.

**Анализ предметной области, создание и обновление тезауруса** выполняются специалистом-индексатором с привлечением нормативных и справочных материалов, языка ЯРТ и программ А и Х.

Индексатор анализирует потенциальное множество документов, которые будут храниться и обрабатываться в системе, и совокупность возможных запросов на поиск в этом множестве документов, с которыми абоненты будут обращаться к системе (рис. 1). В результате такого анализа он выделяет основные понятия, выражающие существенные признаки различных классов документов, а также определяет ключевые слова и словосочетания, представляющие эти понятия в текстах документов.

Затем индексатор заменяет ключевые слова дескрипторами, приводя их к системному формату. Дескриптор в системном формате — это строка длиной не более 20 литер, не начинающаяся с пробела и не содержащая запятой и точки с запятой. Среди группы дескрипторов, которые описывают одно и то же понятие, выделяется главный дескриптор (предпочтительный термин). Остальные дескрипторы группы называются синонимами. Каждому понятию, а следовательно, и дескрипторам, его описывающим, присваивается уникальный пятизначный номер (НДР) от 00001 до 99999. НДР является внутрисистемным дескриптором этого понятия.

Система АСПИД-3 воспринимает в поисковых образах документов и запросов любой дескриптор данного понятия — главный дескриптор, синонимы, НДР, но в большинстве случаев для дальнейшей работы и поддержки сохраняет и документирует только НДР и главный дескриптор, т. е. выполняет «принуждение к предпочтительному термину». Применение НДР вызвано соображениями эффективности внутрисистемного представления и обработки, а также возможностью стенографической ссылки на понятие во внешнем представлении.

Для подразделения на систематические классы каждому понятию может быть присвоен однолитерный код (КП). Для подразделений дескрипторов на классы, например термины, записанные в русском и латинском алфавитах, каждому дескриптору также может быть присвоен код (КД).

Далее индексатор анализирует основные парадигматические (т. е. внутренне присущие и не зависящие от контекста) связи между понятиями предметной области: родо-видовые, соподчинения, причины-следствия, сходства, ассоциации, части-целого и др.

Различают симметричные связи (например, если А сходно с Б, то и Б сходно с А) и несимметричные (например, если А причи-

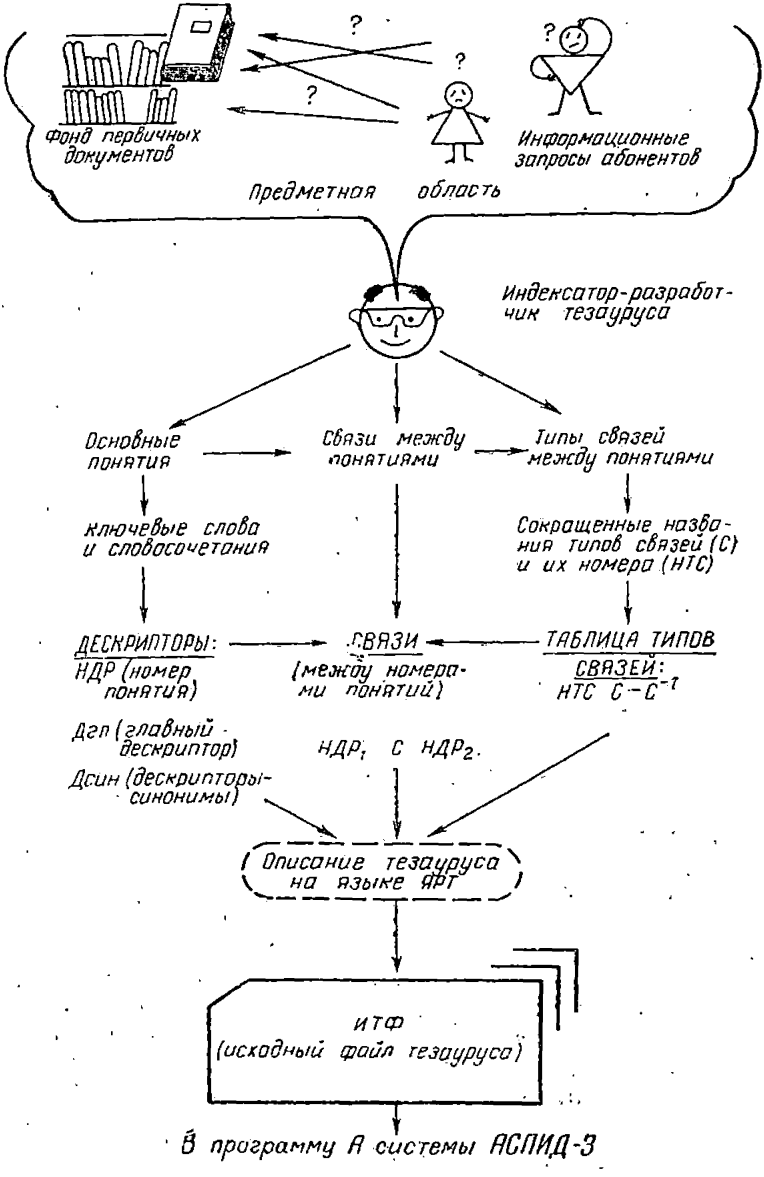



Рис. 1. Схема разработки тезауруса для ввода в систему АСПИД-3

на Б, то Б следствие А). Выделенным типам симметричных связей даются сокращенные четырехлитерные названия (С), например СХОД, а несимметричных — два названия для указания названия связи в прямом и обратном направлениях ($С — С^{-1}$), например ПРИЧ-СЛЕД. АСПИД-3 допускает регистрацию в тезаурусе до 50 типов связей. Типы связей нумеруются (НТС) для указания их упорядоченности при распечатке тезауруса и заносятся в таблицу типов связей (ТТС).

Затем индексатор регистрирует конкретные связи между понятиями, включаемыми в тезаурус, указывая номера понятий и название связи между ними, причем достаточно задать название связи лишь в одну сторону. Название и указание связи в обратном направлении (инверсия связи) будут установлены системой автоматически.

Имя создаваемого тезауруса, НДР, главные дескрипторы, дескрипторы-синонимы, коды понятий и дескрипторов, таблица типов связей с номерами и названиями связей, связи между понятиями записываются на бланках кодирования в виде операций символического языка работы с тезаурусом ЯРТ и переносятся на машинный носитель, образуя исходный файл тезауруса (ИТФ).

**Программа А** в режиме создания читает ИТФ (рис. 2) и создает внутрисистемный тезаурус, состоящий из файла тезауруса (ТФ) и файла связей тезауруса (СФ), располагающихся на магнитных дисках.

Принятая структура файлов ТФ и СФ обеспечивает достаточно компактное внутреннее представление тезауруса, эффективную обработку и преобразование из внешнего во внутреннее представление и обратно, а также возможность реализации некоторых процедур автоматической защиты и поддержки целостности тезауруса, средств помощи индексатору документов и запросов и средств поддержки файла документов.

Файл тезауруса состоит из НД- и ДН-частей и двухуровневых индексов к этим частям. НД-часть («НД-Дескриптор») содержит список НДР и главных дескрипторов всех понятий тезауруса с указанием кодов понятий. Индекс обеспечивает быстрый поиск для проверки наличия заданного НДР в тезаурусе и выдачу главного дескриптора по заданному НДР. ДН-часть («Дескриптор — НДР») включает список всех дескрипторов тезауруса (главных и синонимов) с соответствующими НДР и кодами дескрипторов. С помощью индекса обеспечивается быстрый поиск для проверки наличия заданного дескриптора в тезаурусе и выдачу НДР по заданному дескриптору.

Файл связей тезауруса состоит из ТС- и НС-частей. ТС-часть («Таблица типов Связей») содержит список всех фиксируемых в данном тезаурусе типов связей и служит для проверки наличия заданного типа связи, определения инверсии для несимметричной связи, и преобразования названия связи из внешнего представления во внутрисистемный код и обратно. НС-часть («НДР — Связи») включает список всех связей, установленных между понятиями, вклю-

ченными в тезаурус, с указанием номеров понятий НДР и кодов типов связей между ними.

В режиме обновления программа $A_2$ читает файл обновлений тезауруса (ФОТ) и «старые» файлы тезауруса ТФ и связей СФ и строит «новый» тезаурус — файлы ТФ(Н) и СФ(Н).

Внутрисистемное представление тезауруса может быть преобразовано программой X во внешнее представление в виде распеча-

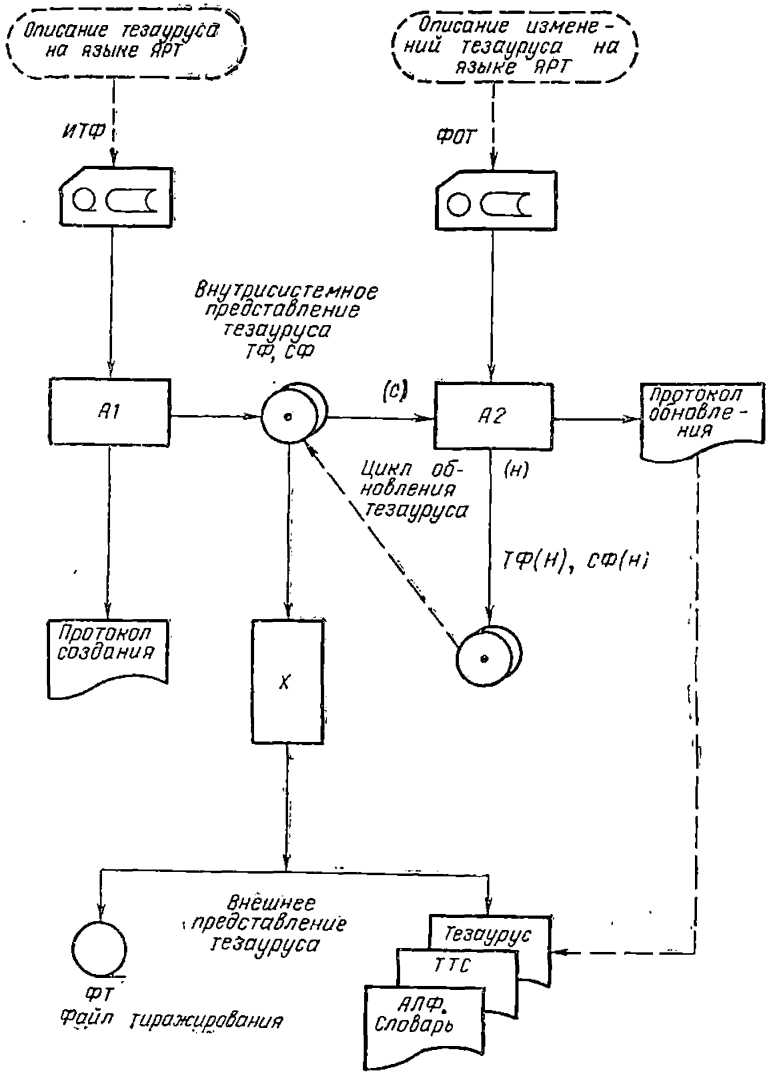



Рис. 2. Схема создания и поддержки тезауруса в системе АСПИД-3

ток алфавитного словаря дескрипторов, таблицы типов связей и собственно тезауруса.

**Алфавитный словарь дескрипторов** содержит список дескрипторов (главных и синонимов) тезауруса с указанием НДР соответствующих понятий и включает латинскую и русскую части. В латинскую часть выделяются все дескрипторы тезауруса, имеющие код дескриптора <кд>=«Л». Дескрипторы этой части словаря отсортированы по латинскому алфавиту. Все остальные дескрипторы относятся к русской части и сортируются по русскому алфавиту.

**Таблица типов связей** содержит перечень всех заданных в тезаурусе типов связей с указанием номера и сокращенного названия типа связи для симметричных связей либо пары сокращенных названий прямой и инверсной связи для несимметричных связей. Перечень упорядочен по номерам типов связей.

**Тезаурус** распечатывается по **гнездам понятий.** Структура гнезда понятия во внешнем представлении тезауруса приведена ниже:

<ндр> <дгл> (<кд>) <кп>
<дсин$_1$> (<кд$_1$>)
.
.
.
<дсин$_n$> (<кд$_n$>)
$c_1$ <ндр$_1$> <дгл$_1$>
.
.
.
<$c_m$> <ндр$_m$> <дгл$_m$>,

где <ндр> — номер понятия (внутрисистемный дескриптор этого понятия);
<кп> — семантический код понятия;
<дгл> — главный дескриптор понятия (предпочтительный термин);
<дсин$_i$> — дескриптор-синоним понятия;
<кд> — код дескриптора;
<$c_i$> — название связи между данным понятием и понятием под номером <ндр$_i$> в направлении к данному понятию.

Таким образом, для каждого понятия в гнезде указан НДР понятия, его код, главный дескриптор, синонимы (с их кодами) и связи с другими понятиями. Для каждой связи указывается название типа связи, НДР и главный дескриптор понятия, с которым установлена связь.

Гнезда понятий в тезаурусе располагаются упорядоченно по возрастанию их НДР, синонимы в гнезде — по возрастанию кодов литер, связи — по возрастанию номеров типов связей. Для удобства поиска незанятых («вакантных») в данном тезаурусе НДР места возможных вставок гнезд помечены в распечатке звездочкой.

Принятые организация и форматы распечаток обеспечивают достаточную эффективность ручной работы с тезаурусом и простоту написания операций обновления тезауруса.

Тезаурус служит нормативным словарем-справочником для системы и ее абонентов и дает унифицированное толкование предметной области. Дескрипторы и/или НДР, включенные в тезаурус, используются при формировании поисковых образов документов и поисковых образов запросов.

Отметим основные отличия внешнего представления тезауруса, принятого в системе АСПИД-3, от традиционного:

дескрипторные статьи заменены гнездами понятий;

неинформативные и фактически дублируемые в алфавитных словарях статьи дескрипторов-синонимов удалены из тезауруса, что позволяет сократить его объем и устранить излишние отсылки на основные термины;

семантически ненагруженный алфавитный порядок дескрипторных статей заменен последовательностью гнезд понятий в порядке возрастания номеров понятий, что позволяет присваивать номера понятий в соответствии с некоторой систематической классификацией, располагать гнезда понятий в систематическом порядке и в ряде случаев устранять необходимость в систематическом указателе к тезаурусу;

возможность использования номеров понятий во внешнем представлении позволяет применять стенографический стиль при индексации документов и запросов и в операциях обновления тезауруса.

### Операции языка работы с тезаурусом ЯРТ

1. **Идентифицировать тезаурус:**

ИДТ{<итн> / <итс> , <втс> [,<ито>]}

Присваивает создаваемому «новому» тезаурусу идентификатор <итн>, нулевой номер версии и дату создания; идентифицирует обновляемый «старый» тезаурус, указывая его идентификатор <итс>, и номер версии <втс>; обновленному тезаурусу присваивает либо <итс>, <втс>+1 и дату обновления, либо может заменить его идентификатор на <ито>, присвоить нулевой номер версии и дату обновления.

2. **Создать гнездо:**

НДР + <ндр> <дгл> [,Д<кд>] [,Н<кп>]

В тезаурусе создается новое гнездо понятия с номером <ндр>, главным дескриптором <дгл>, кодом дескриптора <кд> и кодом понятия <кп>. По умолчанию <кд> и <кп> — пробелы.

3. **Обновить главный дескриптор:**

НД: <ндр> <дгл> [,Д<кд>] [,Н<кп>]

В гнезде с номером <ндр> значения <дгл>, <кд> и <кп> заменяются указанными в операции.

4. **Заменить главный дескриптор:**

НД=<ндр> <дсин>

В гнезде с номером <ндр> бывший главный дескриптор делается дескриптором-синонимом, а указанный синоним <дсин> становится главным дескриптором.

5. **Вставить синоним:**

ДН+<ндр> <дсин> [,Д<кд>]

В гнездо с номером <ндр> вставляется дескриптор-синоним <дсин> с кодом <кд>.

6. **Обновить код дескриптора:**

ДК : <ндр> <д> [,Д<кд>]

В гнезде с номером <ндр> у указанного дескриптора <д> код заменяется на <кд>.

7. **Удалить синоним:**

ДН — <ндр> <дсин>

Из гнезда с номером <ндр> удаляется указанный дескриптор-синоним <дсин>.

8. **Удалить гнездо:**

НД — <ндр>

Из тезауруса исключаются гнездо понятия с номером <ндр>, все дескрипторы и все связи этого понятия с другими понятиями.

9. **Вставить тип связи:**

ТС+<нтс>)<с> [—<$с^{-1}$>]

В таблицу типов связей под номером <нтс> заносится указанный тип связи между понятиями с названием связи <с> для симметричной связи либо с названиями <с> и <$с^{-1}$> для несимметричной связи (<с> и <$с^{-1}$> — название связи в прямом и обратном направлениях соответственно).

10. **Обновить названия типа связи:**

ТС : <нтс>)<с> [—<$с^{-1}$>]

В таблице типов связей у типа связи с номером <нтс> названия связей будет заменены указанными. В гнездах тезауруса произойдет аналогичная замена названий связей у всех связей этого типа.

11. **Удалить тип связи и связи этого типа:**

ТС — <нтс>)<с> [—<$с^{-1}$>]

Из таблицы типов связей удаляется тип связи с указанным номером и названием (названиями), из всех гнезд тезауруса удаляются все связи этого типа.

12. **Вставить связь:**

НС+<ндр1> <с> <ндр2>

Понятие с номером <ндр1> связывается с понятием <ндр2> связью с названием <с> в направлении от <ндр2> к <ндр1>.

В гнезде понятия <ндр1> появляется указание связи в виде <с><ндр2><дгл2>, а в гнезде <ндр2> — указание связи <с$^{-1}$><ндр1><дгл1>.

13. **Удалить связь:**

НС—<ндр1> <с> <ндр2>

Из тезауруса удаляется указанная связь между понятиями <ндр1> и <ндр2>.

При создании и обновлении тезауруса программа А контролирует синтаксис операций ЯРТ, правильность идентификации тезауруса, уникальность вводимых в тезаурус НДР и дескрипторов, правильность типов связей и связей между понятиями (уникальность номеров и названий связей в таблице типов связей, уникальность связей и наличие связываемых гнезд), достраивает инверсии связей, проверяет корректность обновлений.

Программа Х сопровождает распечатку тезауруса идентификацией тезауруса, названием части тезауруса, нумерацией, индексом страниц и статистикой с указанием количеств гнезд понятий, дескрипторов, дескрипторов-синонимов и связей каждого типа.

## СОЗДАНИЕ ФАЙЛА ДОКУМЕНТОВ

В системе АСПИД-3 основной единицей хранения, поиска и выдачи информации является документ. Прообразами накапливаемых в системе документов служат некоторые внешние первичные документы или сообщения, например книги, статьи, отчеты, библиографические карточки, таблицы, анкеты, распоряжения и т. п.

Способы отображения первичных документов в системные могут быть различными и определяются назначением и общей организацией информационно-поисковой системы. В зависимости от этого один системный документ может быть образом одного первичного, нескольких первичных или какой-то части одного первичного документа. В отдельных случаях отображение может выполняться автоматически специальной программой пользователя. Чаще всего эта работа возлагается на специалиста-индексатора документов, который анализирует содержание первичных документов и в соответствии с принятым способом отображения описывает формируемые на их основе системные документы операциями языка работы с документами ЯРД.

Каждому создаваемому системному документу индексатор присваивает абсолютный либо относительный номер документа (НДК), который уникально идентифицирует данный документ в множестве всех документов системы, либо среди документов, одновременно используемых в каком-нибудь процессе обработки. Например, относительный номер документа в порции вновь поступивших документов. Далее индексатор планирует содержимое поискового образа документа (ПОД) и хранимого текстового образа документа (ХОД).

ПОД — это список дескрипторов из информационно-поискового тезауруса, которые отражают основное содержание данного документа с точки зрения информационных потребностей потенциальных абонентов ИПС и используются при поиске.

ХОД — это текстовая часть системного документа, которая может выдаваться целиком или частично в ответ на запрос абонента. В зависимости от типа ИПС ХОД может содержать, например, полный текст первичного документа или его реферат, или только библиографические данные, адрес хранения первичного документа или адрес хранения некоторого вторичного документа в другом контуре ИПС.

ХОД записывается строками по 60 литер и может иметь от нуля до 999 строк. Каждая строка сопровождается однолитерным кодом типа строки. Этот код управляет селективной выдачей требуемых частей ХОД, если в запросе абонента было указано, строки каких типов следует включать в ответ или исключать из ответа.

В частных случаях применения, а также на отдельных технологически целесообразных этапах подготовки законченного системного файла документов, ПОД либо ХОД документа (но не оба одновременно) могут отсутствовать либо добавляться к документу позднее и по частям.

На рис. 3 показана общая схема обработки очередной порции первичных документов ΔД, поступивших в фонд первичных документов. Индексатор документов, руководствуясь принятым в ИПС способом отображения, содержимым документов и тезаурусом, записывает совокупности НДК, ПОД и ХОД на бланках кодирования в виде операций ЯРД. Эти описания, перенесенные на перфо- или магнитный носитель, образуют исходный файл документов (ИДФ). Программа В в режиме создания ($B_1$) читает ИДФ и создает системный файл документов (ДФ). При этом она использует файл ТФ для проверки наличия в тезаурусе дескрипторов, включаемых в поисковые образы документов, и замены их соответствующими НДР.

Если в системные документы, хранимые в файле ДФ, требуется внести изменения или исправления, эти корректировки также записываются в виде операций ЯРД и переносятся на перфо- или магнитный носитель, образуя исходный файл обновлений документов (ФОД). Программа В в режиме обновления ($B_2$) читает «старый» (обновляемый) ДФ, читает ФОД, в необходимых случаях обращается к файлу ТФ тезауруса, выполняет требуемые изменения и выдает «новый» (обновленный) ДФ, обозначенный на схеме как ДФ(Н).

Процедура обновления может повторяться многократно. При этом ДФ(Н) перемещается по схеме на место ДФ («цикл обновления»: ДФ(Н) → ДФ → В2 → ДФ(Н)).

С помощью операции выделения документов требуемые документы файла ДФ могут быть выделены в отдельный файл выделенных документов (ДФЫ), который может далее обрабатываться как

любой системный файл документов (ДФЫ→ДФ или ДФЫ→→ДФ(Н)).

Для ускорения последующего поиска документов программа В может создавать для файла ДФ двухуровневый индекс ДФИ, а для последующего обновления поискового файла программой С2 — файл обновлений поискового файла (ФОП).

Программой *У* можно распечатать содержимое файла ДФ либо проверить соответствие ПОД системных документов тезаурусу.

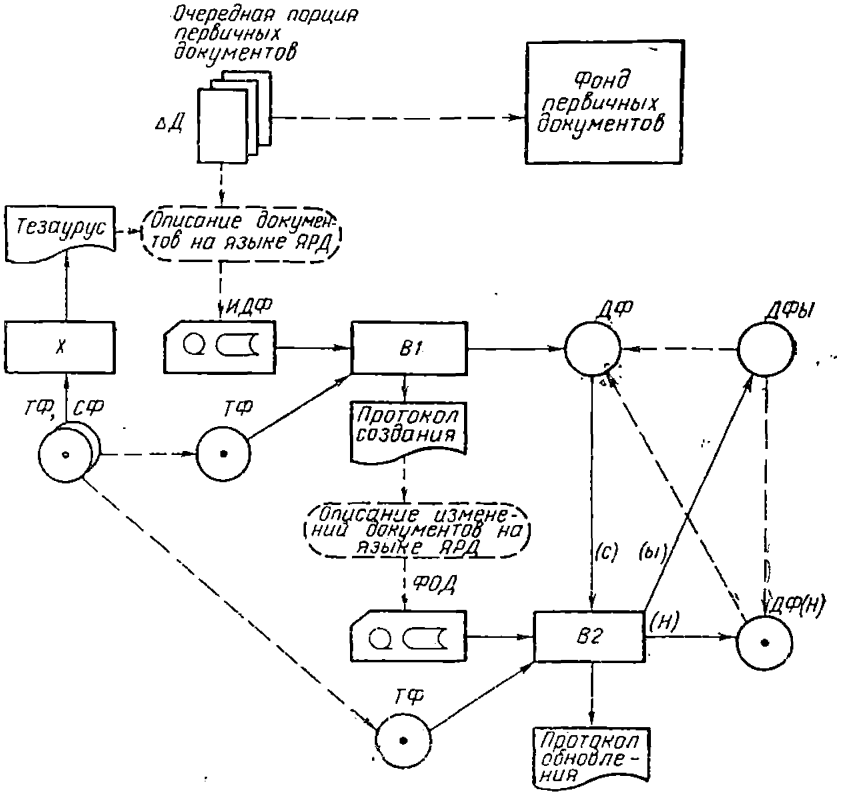



Рис. 3. Обработка очередной порции поступивших документов

## ЯЗЫК РАБОТЫ С ДОКУМЕНТАМИ

Язык работы с документами ЯРД служит для описания системных документов при создании файла документов ДФ и для описания изменений и исправлений, вносимых в файл ДФ при обновлении.

Операции ЯРД записываются в гибком формате, который не препятствует применению в специальных случаях разнообразных жестких форматов для кодирования. Каждая операция начинается с ключевого буквосочетания — мнемонического кода операции, за

которым могут следовать модификаторы и параметры, разделяемые литерами-разделителями, и оканчивается ограничителем «точка с запятой». В одной строчке бланка кодирования, соответствующей одной логической записи файла ИДФ/ФОД, должно размещаться целое число операций ЯРД. Каждая операция может начинаться с произвольной позиции, перед и между операциями может присутствовать произвольное число пробелов. Последовательность операций и записей в файле произвольная, кроме операций указания нестандартных режимов работы. Если эти операции требуются, они должны размещаться в первой записи файла ИДФ/ФОД. Признаком присутствия такой первой записи служит буква Р («Режимы») в ее первом байте.

**Метапеременные, используемые в форматах операций ЯРД.**

1) <ндк> — номер документа, число от 1 до 999999;
2) <д> : : =0/<ндк> — указатель объектных документов операций, если задан 0 — операция осуществляется над всеми документами файла, если <ндк> — над документом с этим номером, если <д> опущен — над объектным документом предыдущей операции в той же строке бланка кодирования;
3) <список д> : : =<элемент списка д>/<список д>, <элемент списка д>
   <элемент списка д> : : =<д>/<ндк> : <ндк>
   Список объектных документов; диапазон <ндк> : <ндк> — указывает документы, номера которых лежат в этом диапазоне;
4) <ндр> — номер понятия, число от 1 до 99999;
5) <список ндр> : : =
   <элемент списка ндр>/<список ндр>, <элемент списка ндр>
   <элемент списка ндр> : : ={+/—}<ндр>/{+/—}<ндр> : : <ндр>
   Список номеров понятий, включаемых (если +) в поисковый образ или исключаемых (если —) из него. Если задан диапазон номеров <ндр> : <ндр>, то в режиме работы без тезауруса включаются/исключаются все номера понятий в этом диапазоне; в режиме работы с тезаурусом исключаются все, а включаются только присутствующие в тезаурусе;
6) <дескриптор> — дескриптор, литерал от 1 до 20 любых литер, кроме первого пробела, запятой и точки с запятой;
7) <список др.> : : =<элемент списка др>/<список др>, <элемент списка др>

<элемент списка др> : : = {+/—} <дескриптор>
Список дескрипторов (главных либо синонимов), включаемых в ПОД (если +) или исключаемых (если —) из него;

8) <строка> — строка документа, последовательность от 0 до 60 произвольных литер, кроме двух подряд идущих знаков «процент»;

9) <т> — код типа строки в документе, любая литера, кроме !/?/:/,/;

10) <с> — указатель номера строки в документе; либо число от 1 до 999, либо 0 или *, означающие фиктивные «нулевую» и «последнюю» строки документа соответственно, для вставок перед первой и после последней строки;

11) <в> — смещение во вставке относительно строки, указанной значением <с> в этой же операции, число от 0 до 999;

12) <список с> : : = <элемент списка с>/<список с>, <элемент списка с>
<элемент списка с> : : = <с>/<с> : <с>
Список номеров строк документа, диапазон <с> : <с> указывает на строки с номерами в этом диапазоне;

13) <список т> : : = <элемент списка т>/<список т>, <элемент списка т>
<элемент списка т> : : = <т>/<т><т>
Список типов строк. Однолитерные элементы <т> и первые литеры двулитерных <т><т> указывают, каких типов строки будут оставлены в обновленном документе или удалены из него. Вторые литеры указывают замену типа с первой литеры на вторую у попадающих в документ строк.

**Операции языка ЯРД**

**Операции указания режимов работы** задают режимы работы, отличающиеся от стандартных. Все эти операции, если они необходимы, должны быть записаны в первой логической записи файла ИДФ/ФОД, в первом байте которой должна стоять буква Р («Режимы»), за которой в произвольном порядке располагаются требуемые из следующих четырех операций:

1. Работать Без файла тезауруса: Б;

Если операция Б не задана, по умолчанию действует стандартный режим работы с использованием файла тезауруса. Если эта операция задана, программа В при создании или обновлении поисковых образов документов будет включать в них номера понятий (НДР) без проверки наличия в тезаурусе понятий под такими номерами. Она также не сможет воспринимать в операциях ЯРД

главные дескрипторы и синонимы и преобразовывать их в НДР. При этом работа программы В ускоряется, а ответственность за правильность кодирования поисковых образов документов номерами понятий возлагается на пользователя или его программные средства автоматической индексации документов.

2. Работать с файлом ДФЫ: Ы;

Если эта операция задана, программа В будет выделять из обновляемого файла ДФ документы, указанные в операциях ЦЫ, и помещать их в отдельный файл выделенных документов ДФЫ. По умолчанию файл ДФЫ не создается и операции выделения документов ЦЫ не будут выполняться.

3. Печатать подробный Листинг: Л;

Если эта операция задана, при создании и обновлении файла ДФ в протоколе будет проводиться подробная печать всех затронутых частей документов: в ПОД — вставленные и удаленные НДР и главные дескрипторы, в ХОД — созданные или модифицированные части.

По умолчанию для сокращения объема протокола осуществляется только печать принятых к исполнению исходных операций ЯРД, затрагивающих весь файл (но не результатов их действий!), и исходных локальных операций. Информация об изменениях номеров и типов строк в результате исполнения локальных операций также не печатается.

4. СМестить или сЖать номера документов:

{М {+/—}/Ж+} <ндк>;

Если задан вариант смещения М+<ндк>, номера документов, включаемых в файл обновленных документов ДФ(Н), увеличиваются на значение <ндк>. Если задан вариант М—<ндк>, происходит уменьшение номеров документов на значение <ндк>. Если задан вариант сжатия Ж+<ндк>, документам файла ДФ(Н) присваиваются новые последовательно возрастающие на 1 номера, начиная с номера <ндк> первого в файле ДФ(Н) документа. По умолчанию номера документов не изменяются.

В следующих двух группах операций первая литера мнемонического кода операции указывает часть документа, над которой осуществляется или разрешается операция, вторая — тип операции.

Первая литера:

Ц — документ Целиком;

П — ПОД документа;

Х — ХОД документа.

Вторая литера:

Ы — вЫделить документ в отдельный файл ДФЫ;

В — Вставить;

О — Обновить;

З — Заменить;

У — Удалить.

Объектные документы (операнды) этих операций задаются не-

обязательным списком [<список д>], перечисляющим номера документов, на которые распространяется данная операция. Если список опущен, операндом по умолчанию является объектный документ предыдущей операции, записанной в той же строке бланка кодирования. Если в качестве номера документа в списке задан 0, операция распространяется на все документы файла. Если элементом списка указан диапазон номеров документов <ндк> : <ндк>, то операция распространяется на все документы файла, номера которых лежат в этом диапазоне.

**Операции выделения и удаления**

5. ВЫделить Целые документы в файл ДФЫ

ЦЫ [<список д>];

Если был указан режим работы с файлом ДФЫ, то документы файла ДФ, перечисленные в <списке д>, будут выделены в отдельный файл ДФЫ и исключены из обновленного файла ДФ(Н). Никакие другие операции над выделяемыми документами не выполняются, и документы включаются в файл ДФЫ в том виде, в котором они присутствовали в файле ДФ.

6. Удалить Целые документы, или Под, или Ход

{Ц/П/Х/} У [<список д>];

Указанные <списком д> документы исключаются из файла целиком (вариант ЦУ) либо исключаются их поисковые образы (вариант ПУ), либо хранимые текстовые образы (вариант ХУ).

**Операции, разрешающие обработку документов или их частей,** указывают типы разрешенных операций обработки документов и список номеров документов, предназначенных для обработки. Могут использоваться администратором системы в качестве механизма защиты или для ограничения действия ошибок кодирования и перфорации.

7. Разрешить Вставить (создать) новые документы

ЦВ [<список д>];

<Список д> указывает номера новых документов, которые разрешается создать или вставить при обновлении в файл документов.

8. Разрешить Обновить или Заменить Целые документы, или ПОД, или ХОД

{Ц/П/Х/} {О/З} [<список д>];

<Список д> указывает номера документов, которые разрешается затрагивать соответствующей обработкой. Если указана замена (З), то объектный документ целиком (если Ц), или его ПОД (если П), или ХОД (если Х) удаляются и операции из файла ФОД создают новый документ (или его соответствующую часть) под тем же номером.

**Операции обработки поискового образа документа**

9. Включить/Исключить НДР

Н [<д>] <список ндр>;

Эта операция позволяет включать (если в соответствующем элементе списка указан «+») в поисковый образ документа или исключать из него (если «—») номера понятий, заданные <списком ндр>. Если задан диапазон исключаемых номеров <ндр>: :<ндр>, то исключаются все НДР, находящиеся в этом диапазоне. Если задан диапазон включаемых номеров, то в режиме работы без тезауруса в ПОД включаются все НДР этого диапазона. В стандартном режиме работы с тезаурусом происходит проверка указанного включаемого диапазона, и операция в этом диапазоне выполняется только в том случае, если все НДР диапазона присутствуют в тезаурусе.

Суммарно в списках операций Н0, распространенных на весь файл, может быть указано не более 10 элементов.

10. Включить/исключить Дескрипторы

Д [<д>] <список др> ;

Эта операция выполняется только в стандартном режиме работы с тезаурусом. В качестве включаемого в ПОД дескриптора (если задан «+») или исключаемого из него (если задан «—») может быть указан главный дескриптор или любой синоним, присутствующие в тезаурусе.

Суммарно в списках операций Д0, распространенных на весь файл, может быть указано не более 10 дескрипторов.

**Операции обработки хранимого текстового образа документа**

11. Создать, заменить или вставить строку

+ <т> ) <строка> % %[<д>].<с>[.<в>];

В режиме создания ДФ(Н) эта операция задает <строку> текста ХОД с кодом типа строки <т>, которая должна быть создана под порядковым номером строки <с> в документе с номером [<д>]. Если в строке менее 60 литер, она будет дополнена справа до длины 60 литер пробелами.

Под созданием *i*-й строки документа автоматически подразумевается создание всех предыдущих строк, и если они не были заданы явно, система порождает на их месте пустые (состоящие из пробелов) строки с кодом типа строки «пробел».

Если в режиме обновления на ХОД объектного документа распространяется действие операции ЦЗ или ХЗ, разрешающей замену ХОД, весь старый ХОД документа уничтожается и заменяется строками, заданными в операциях замены строк файла ФОД.

Если в режиме обновления на ХОД объектного документа распространено действие операции ЦО или ХО, разрешающей обновление ХОД, и <с> указывает на уже существующую в ХОД строку, происходит ее замена на заданную в этой операции.

Если <с> указывает на строку за пределами ХОД обновляемого документа, происходит создание новой строки (и возможно, нескольких пустых, предшествующих ей).

Конструкция <с>·<в> позволяет вставлять одну или несколько новых строк между строками обновляемого или создаваемого документа. При этом <с> указывает строку, за которой осуществляется вставка, а <в> — номер строки во вставке. Если <с>=0, вставка производится перед первой, а если <с>=* — вслед за последней строкой документа.

Строки вставки нумеруются последовательно, начиная с 1. При вставке *i*-й строки автоматически порождаются и заполняются пробелами не указанные явно предыдущие строки вставки.

Суммарное число строк в документе, считая и все строки вставок, не должно превышать 999. При каждом обновлении строки документа перенумеруются в порядке их следования, начиная с 1.

12. Удалить строки

—[<д>]·<список с>;

В результате выполнения этой операции из указанного документа (или всех документов файла, если <д>=0) удаляются строки с номерами, заданными списком номеров строк.

13. Заменить код типа Строк

С <т>) [<д>]·<список с>;

В результате этой операции в документе [<д>] в строках, заданных списком номеров строк <список с>, код типа строки заменяется указанным кодом <т>.

14. Выделить части документа по кодам Типов строк и заменить коды

Т[<д>]{+/—} <список т>;

В варианте формата +<список т> однолитерные элементы <т> и первые литеры двулитерных элементов <т><т> указывают, строки каких типов будут выделены из старого документа и включены в новый. Двулитерные элементы указывают, что при этом произойдет замена кода типа строки, заданного первой литерой, кодом типа, заданным второй литерой.

В варианте — <список т> однолитерные элементы указывают, строки каких типов будут исключены из документа, а двулитерные задают замену кодов типов у оставшихся строк.

Операция Т выполняется после выполнения всех других операций над данным документом. Поэтому, строго говоря, ее объектом является не ХОД «старого» документа, а ХОД «старого» документа после выполнения всех других операций обновления (включая и возможную операцию замены кода типов у строк с заданными номерами строк).

**Инвертированный поисковый файл (ПФ)** создается программой С1, которая читает ПОД из файла ДФ и составляет для каждого НДР список номеров документов, в ПОД которых этот НДР входит. Для ускорения поиска в ПФ по заданному НДР программа С1 строит двухуровневый индекс к файлу ПФ.

Для интегральной характеристики поисковых образов данного файла документов можно построить вспомогательной программой Z

**частотные словари дескрипторов.** Программа Z читает файлы ПФ и ТФ и печатает словари (по убыванию частоты встречаемости НДР в ПОД и в порядке возрастания номеров НДР), указывая для каждого НДР главный дескриптор соответствующего понятия и число вхождений этого НДР в поисковые образы документов данного ДФ.

Частотные словари дескрипторов могут использоваться для корректировки тезауруса, уточнения метода индексации документов или запросов.

**Проведение поиска** требуемых документов по запросам абонентов предполагает в общем случае, что в информационно-поисковой системе средствами пакета прикладных программ АСПИД-3 созданы и поддерживаются основные системные файлы: тезаурус, файл документов и инвертированный поисковый файл.

Модульная структура пакета АСПИД-3, состав и функциональные возможности входящих в него языков и программ позволяют строить на его основе различные варианты ИПС, исключать отдельные компоненты или напротив пополнять систему собственными средствами пользователя и подключать ее к другим системам.

На рис. 4 показана некоторая типовая схема поиска, когда абоненты передают свои неформальные запросы специалисту-индексатору запросов. Индексатор анализирует неформальный запрос, выделяет в нем ключевые слова, с помощью тезауруса заменяет их дескрипторами, составляет формализованные поисковые предписания на языке запросов на поиск ЯЗП, указывает режимы поиска и выдачи ответов. Основой запроса является булевское выражение из дескрипторов, описывающее содержание требуемых документов.

Совокупность запросов, оформленная в виде файла запросов на поиск ФЗП, вводится в программу поиска D. Программа D с помощью поискового файла вычисляет номера документов, поисковые образы которых удовлетворяют булевскому выражению запроса, т. е. обращают булевское выражение в «истину», если дескрипторы выражения, присутствующие в ПОД, заменить значением «истина», а отсутствующие — значением «ложь». Полученный список номеров релевантных документов может быть использован программой D для востребования этих документов из файла ДФ. При этом в соответствии с указаниями в запросе о форматах выдачи могут быть выделены требуемые части документов.

В своей работе программа D может обращаться к тезаурусу для преобразования дескрипторов, указанных в запросах, в НДР и НДР из поисковых образов найденных документов — в дескрипторы. Полученные ответы направляются индексатору для возможной корректировки запросов и передаются абонентам.

Генерация требуемого варианта программы D дает возможность использовать различные носители для входной и выходной информации, выбирать варианты организации файлов, подключать разнообразные пользовательские программы пред- и постобработки. Язык запросов позволяет управлять границами поиска, объемами, содержанием и форматами выдачи ответов. В ответы можно включать по выбору только количество найденных документов, количест-

во и номера найденных документов, требуемые части документов (ПОД, ХОД либо строки ХОД указанных типов). Допускаются поисковые предписания в виде списка номеров требуемых документов.

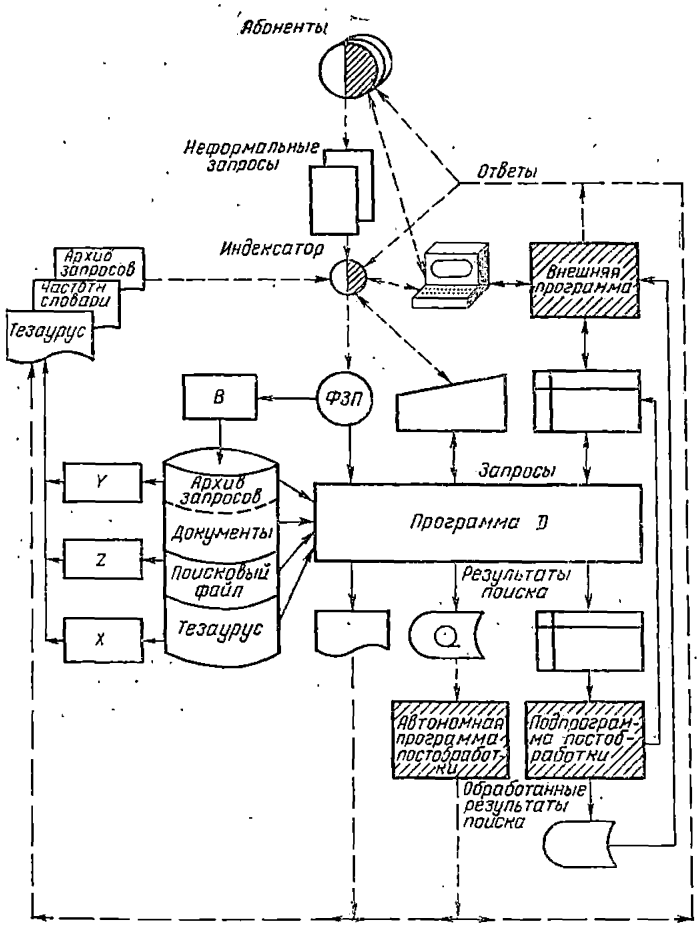



Рис. 4. Схема поиска в системе АСПИД-3

Запросы, имеющие общие подвыражения — термы или общие указания о режимах и форматах поиска и выдачи, можно объединять в серии с выносом общих частей запросов в общую часть серии. Кроме того, система позволяет накапливать типовые термы, запросы и даже серии запросов в архиве и, пользуясь языковыми средствами макровызова, вставлять их в требуемые места потока запросов. Все это сокращает объем работ и при организации избирательного распространения информации о новых поступлениях

в соответствии с профилями интересов постоянных абонентов, и при проведении диалога с системой во время ретроспективного поиска по всему массиву документов.

### Язык запросов

При обращении к АСПИД-3 для поиска требуемых документов пользователь должен формулировать свои запросы в виде последовательности фраз специального языка запросов (ЯЗП). Кроме поискового образа запроса (формулы запроса), каждый запрос может содержать дополнительные указания о границах поиска, форматах выдачи найденных документов и др.

Запросы, имеющие общие подвыражения (термы) в формулах или общие параметры режима поиска и выдачи, объединяются в серию запросов. Серия запросов является наименьшей законченной единицей работы, воспринимаемой и исполняемой программой поиска, и состоит из общей части и запросов. Отдельный запрос вне серии системой не воспринимается.

Одна или несколько серий запросов, перенесенные на машинный носитель, образуют файл запросов на поиск (ФЗП). ФЗП может быть в различных форматах (ПК-, МД- или ДФ-формате) либо представлять собой поток логических записей неопределенной длины, вводимых через пишущую машинку или из вызывающей программы через область оперативной памяти.

Общим для всех форматов ФЗП является наличие в каждой логической записи двух полей, в которых размещаются фразы языка:

П1 — однобайтового поля, в котором размещается код типа фразы — первая литера соответствующей фразы;

П2 — 60-байтового поля, в котором размещается текст фразы (в записях неопределенной длины это поле имеет длину не более 60 байт).

Одна фраза занимает одну логическую запись или несколько последовательных записей. У всех записей одной и той же фразы и у однотипных фраз в поле П1 должен быть указан один и тот же тип фразы. При переходе продолжения фразы в следующую запись не допускаются разбиение и перенос в следующую запись частей номеров термов, номеров документов, НДР и дескрипторов.

### Структура файла запросов

```
<файл запросов> ::= {<серия запросов>}...
<серия запросов> ::= <общая часть серии>
                     [<запрос>]...
<общая часть серии>::=<заголовок серии>
                     [<режим>]...
                     [<терм>]...
<запрос> ::=          <заголовок запроса>
                     [<режим>]...
                     [<терм>]...
                     [<формула запроса>]
```

<комментарий> — фраза, текст которой воспроизводится при распечатке в месте ее появления в файле запросов;

<вызов запросов> — фраза вызова запросов из архива; может замещать любую фразу или часть файла запросов, или весь файл целиком.

**Заголовок серии** — указывает начало и имя серии запросов.

<заголовок серии> : := С <имя серии> ;

Имя серии при распечатке включается системой в указатель начала серии и повторяется при распечатке указателя начала каждого запроса этой серии.

**Заголовок запроса** — указывает начало и имя запроса.

<заголовок запроса> : := З <имя запроса> ;

Имя запроса включается системой в указатель начала запроса и сопровождается именем серии запросов и датой обработки запроса.

**Режим** — задает параметры, управляющие режимом поиска и выдачи документов.

<режим> : := Р <список параметров>

Параметры следуют в списке в произвольном порядке и разделяются произвольным числом пробелов

<параметр> : := <имя параметра> = <значение параметра> ;

где <имя параметра> : : = {З/Г/К/В/Ф/Х}.

Параметру, опущенному в режиме серии, присваивается значение «по умолчанию», а параметру, опущенному в режиме запроса, — значение соответствующего параметра для серии.

**Параметр З** управляет форматом воспроизведения текстов запросов в файле результатов.

Формат : З = {0/З/Д/Т} ;

Если З=0; в файл результатов выводятся только заголовки серий, заголовки запросов, установленные режимы и комментарий, расположенный вслед за заголовками, но до параметра З=0;.

Если З=З; — выводится полный текст запросов.

Если З=Д; — выводится полный текст запросов и в терминах для каждого заданного дескриптора указывается соответствующий НДР, а для каждого НДР — соответствующий главный дескриптор.

Если З=Т; — выводится то же, что и в случае З=Д; и дополнительно каждая запись-сегмент терма сопровождается указанием числа документов, удовлетворяющих формуле запроса, равной текущей части данного терма (от первого сегмента терма по текущий сегмент).

По умолчанию принимается значение З=Т;.

**Параметр Г** управляет границами поиска в файле ДФ (диапазоном номеров документов, среди которых будет вестись поиск).

Формат: Г=<нг>:<вг>;

где <нг> и <вг> — нижняя и верхняя границы диапазона поиска, двузначные числа от 00 до 40, такие, что выражения 25000·<нг> и 25000·<вг> — 1 дают соответствующие границы поиска, выраженные в номерах документов.

По умолчанию принимается значение Г=00:01;, т. е. поиск ведется в диапазоне номеров документов от 0 до 24999.

**Параметр К** управляет объемом включаемых в ответ сведений в зависимости от числа найденных документов.

Формат: К=<н>;

где <н> — пороговое число найденных документов, число от 0 до 999999.

Если число найденных документов N≤<н>, то в ответ включается количество найденных документов и их содержимое в соответствии со значениями параметров В, Ф и Х. Если N><н>, то при формуле запроса в виде булевского выражения в ответ включается лишь количество найденных документов.

Если количество документов, затребованных в формуле запроса в виде списка номеров документов, превышает К=<н>, первые К документов будут выданы в соответствии с параметрами Ф, Х, об остальных будет сообщено только их количество.

По умолчанию принимается значение К=10.

**Параметр В** управляет выдачей и форматом временных (промежуточных) ответов.

Формат: В={0/К/Ф};

Если В=0;, временные ответы не выдаются; если В=К; — во временных ответах выдается количество документов, найденных при выполнении соответствующей части формулы запроса; если В=Ф; — в первом временном ответе, удовлетворяющем количественным ограничениям, установленным параметром К, документы выдаются в формате, предписанном параметром Ф, а в остальных (включая и окончательный ответ) — только количество документов и их номера.

По умолчанию принимается значение В=К;.

**Параметр Ф** управляет форматом выдачи найденных документов.

Формат: Ф={Н/П/Д/Х/ПХ/ДХ};

Параметр Ф действует при условии, если количество найденных документов N≤<н>.

Если Ф=Н;, в ответ выводятся только номера найденных документов; если Ф=П; — номера и ПОД в виде списка НДР; если Ф=Д; — номера и ПОД в виде списка НДР и главных дескрипторов; если Ф=Х; — выводятся номера и ХОД; если Ф=ПХ; — ПОД в виде НДР и ХОД; если Ф=ДХ; — ПОД в виде НДР и главных дескрипторов и ХОД.

По умолчанию устанавливается значение Ф=ДХ;.

**Параметр X** управляет селективной выдачей строк ХОД в зависимости от кода типа строк.

Формат: X={+/—}(<список типов строк>);

где <список типов строк> — перечень литер кодов типов строк, разделяемых запятыми.

Параметр X действует, если выдача ХОД разрешена параметром Ф=Х; Ф=ПК; или Ф=ДХ;.

Вариант X=+(<список типов строк>); разрешает включение в ответ только строк ХОД, тип которых перечислен списком.

Вариант X=—(<список типов строк>); запрещает включение в ответ строк ХОД, тип которых перечислен списком.

По умолчанию выдаются все строки ХОД.

**Терм** — простое булевское выражение: до десяти дескрипторов или НДР, связанных одной и той же логической операцией «И» либо «ИЛИ», входящее в качестве компонента в формулу запроса.

Формат: <терм>::={<сегмент терма>}...
<сегмент терма>::=<заголовок терма> <тело сегмента>
<заголовок терма>::=Т<номер терма>,{+/&}
<тело сегмента>::={Д=<список дескрипторов>/Н=<список ндр>}

где Т — код типа фразы в поле П1;

<номер терма> — число от 00 до 99, уникальный идентификатор терма в пределах данного запроса (откуда следует, что термы общей части серии должны иметь уникальные номера в пределах данной серии);

& — знак конъюнкции (логическое «И»);

\+ — знак дизъюнкции (логическое «ИЛИ»);

<список дескрипторов> — последовательность разделяемых запятыми дескрипторов (замыкающие пробелы в дескрипторах можно опускать);

<список ндр> — последовательность разделяемых запятыми номеров понятий НДР (ведущие нули в НДР можно опускать).

Фраза <терм> может состоять из одного или нескольких сегментов. У всех сегментов одного терма заголовки должны быть идентичными, а тела могут быть разнотипными, но общее число дескрипторов и НДР в них не должно превышать 10.

Интерпретация: терм Т15, &Д=Д1, Д2, Д3 означает булевское выражение Д1&Д2&Д3, которое в формуле запроса будет представлено номером терма 15.

**Формула запроса** — поисковое предписание в виде списков номеров требуемых документов либо в виде булевского выражения, составленного из номеров термов, которому должны удовлетворять поисковые образы найденных документов

<формула запроса>::={Н<список ндк>;}.../
Б<булевское выражение>;

**В формуле запроса в виде списка номеров документов:**

<список ндк>::=<элемент списка ндк>/
<список ндк>,<элемент списка ндк>
<элемент списка ндк>::=<ндк>/<ндк>:<ндк>

где <ндк> — номер требуемого документа, число от 1 до 999999;

<ндк>:<ндк> — диапазон номеров требуемых документов (выдаются все документы, номера которых принадлежат указанному диапазону). Номер справа от двоеточия должен быть не меньше левого.

Фраз с Н-формулой в запросе может быть несколько. Для Н-формулы не имеют смысла и игнорируются в указателе режима параметры Г и В, действуют параметры Ф и Х, а параметр К указывает предельное число документов, выдаваемых на печать. Если количество затребованных в одной фразе документов превышает К=<н>, первые К документов будут выданы в соответствии с параметрами Ф, Х, об остальных будет сообщено только их количество.

**В формуле запроса в виде булевского выражения:**

<булевское выражение> — композиция из номеров термов, бинарных логических операций «И», «ИЛИ», «И НЕ» (обозначаемых соответственно как &, +, —) и круглых скобок, подчиняющаяся следующим правилам:

1) номер терма, входящий в выражение, представляет в нем соответствующую дизъюнкцию или конъюнкцию дескрипторов или НДР и считается термом первого уровня;

2) нельзя дважды использовать в выражении один и тот же номер терма;

3) в выражении можно применять как номера термов общей части серии, так и номера термов данного запроса;

4) терм старшего уровня образуется из термов младших уровней, связанных однотипной логической операцией, при этом каждый входящий в него терм старше первого уровня должен заключаться в скобки;

5) любой терм можно заключать в скобки;

6) в выражении допускается не более двух уровней скобок;

7) выражение является термом (следствие из п. 6: выражение является термом не старше четвертого уровня).

Среди документов, номера которых находятся в диапазоне поиска, отыскиваются документы, поисковые образы которых обращают в истину булевское выражение формулы запроса в результате подстановки в него вместо номеров термов соответствующих дизъюнкций или конъюнкций дескрипторов и замены дескрипторов, присутствующих в ПОД, значением «истина», а отсутствующих — значением «ложь».

**Комментарий** — фраза, текст которой воспроизводится при распечатке в месте ее появления в файле запросов

Формат: <комментарий>::=*<строка>

Фразы комментария могут появляться в произвольном месте

файла запросов. Однако, если они появляются после фразы, содержащей параметр режима печати текста запросов З=0;, их распечатка подавляется до тех пор, пока не будет установлен режим З≠0;.

**Вызов запросов из архива.** Кроме перечисленных фраз ЯЗП с <кодом типа фразы> : : =С/З/Р/Т/Н/Б/* имеется фраза специального назначения с форматом:

<вызов запросов>: :={=<список ндк>;}
<список ндк>: :=<элемент списка ндк>/
<список ндк>,<элемент списка ндк>
<элемент списка ндк>: :=<ндк>/<ндк> : <ндк>

<список ндк> — определяется так же, как и <список ндк> в Н-формуле запроса и указывает номера документов в файле ДФЗ, текст (ХОД) которых (в порядке заданном списком) замещает данную фазу вызова при чтении файла запросов программой D.

Строки этих документов с кодами типов строк, отличающихся от кодов типов фраз ЯЗП, игнорируются. Фразы вызова могут появиться в любом месте ФЗП, замещать его любую часть и даже весь файл целиком, но вызываемый ими текст должен образовывать правильную последовательность фраз ЯЗП. Рекурсивный вызов не разрешается, т. е. вызываемый текст ХОД в свою очередь не должен содержать фраз <вызов запросов>.

В зависимости от требований конкретного применения ДФЗ, из которого осуществляется вызов, может быть либо отдельным файлом документов, хранящим архив типовых термов, запросов и серий запросов, либо физически совпадать с основным файлом документов.

### Организация избирательного распространения информации и ретроспективного поиска

Одной из основных регулярных задач информационного поиска является избирательное распространение информации (ИРИ) о новых поступлениях в фонд документов среди постоянных абонентов ИПС в соответствии с относительно стабильными профилями их интересов. Разновидностью ИРИ может быть периодическая подготовка тематических библиографий по новым поступлениям.

Такие постоянные запросы в системе АСПИД-3 удобно помещать в архивный файл запросов ДФЗ, имеющий формат файла документов или являющийся его частью. При этом файл запросов на поиск может состоять из одной или нескольких фраз вызова запросов из архива.

Система позволяет с помощью программ B и Y корректировать и пополнять ДФЗ в случае обнаружения ошибок индексации, смещения интересов или появления новых абонентов.

В постоянных запросах пользователь может максимально ускорить обработку и сократить выдачу служебных сообщений. Для этого после окончательной отладки постоянных запросов следует:

заменить дескрипторы номерами НДР;

задать режим З=0;

требовать выдачу временных ответов в том случае, если они действительно необходимы;

требовать только нужные части документов.

Процедура ИРИ (рис. 5) выполняется для каждой очередной порции новых документов $\Delta ДФ_i$, поступающей в систему. Программой $C1$ для $\Delta ДФ_i$ создается инвертированный поисковый файл $\Delta ПФ_i$. Затем программой $D$ обрабатываются указанные в ФЗП постоянные запросы из архива. Для поиска в $\Delta ДФ_i$ используется $\Delta ПФ_i$. Результаты поиска могут быть подвергнуты специальной постобработке и затем рассылаются постоянным абонентам.

Файлы $\Delta ДФ_i$ и $\Delta ПФ_i$ после проведения ИРИ добавляются программами В3 и С3 соответственно к кумулятивным файлам $ДФ_к$ и $ПФ_к$ (рис. 6). Возникающие разовые обновления кумулятивных

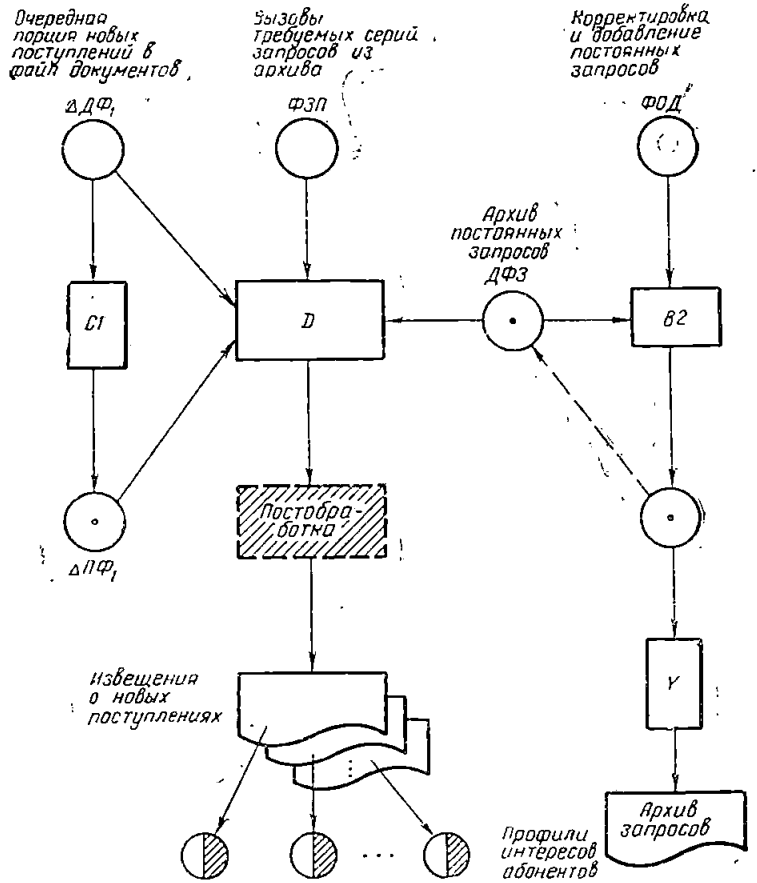



Рис. 5. Избирательное распространение информации

файлов исполняются программами B2 и C2. В случае больших объемов изменений в $ДФ_к$, включающих реорганизацию файла со сменой номеров документов, соответствующий поисковый файл $ПФ_к$ создается вновь программой C1. Ретроспективный поиск информации (РПИ) по разовым запросам во всем массиве хранимых документов выполняется программой *D*, работающей с файлами $ДФ_к$

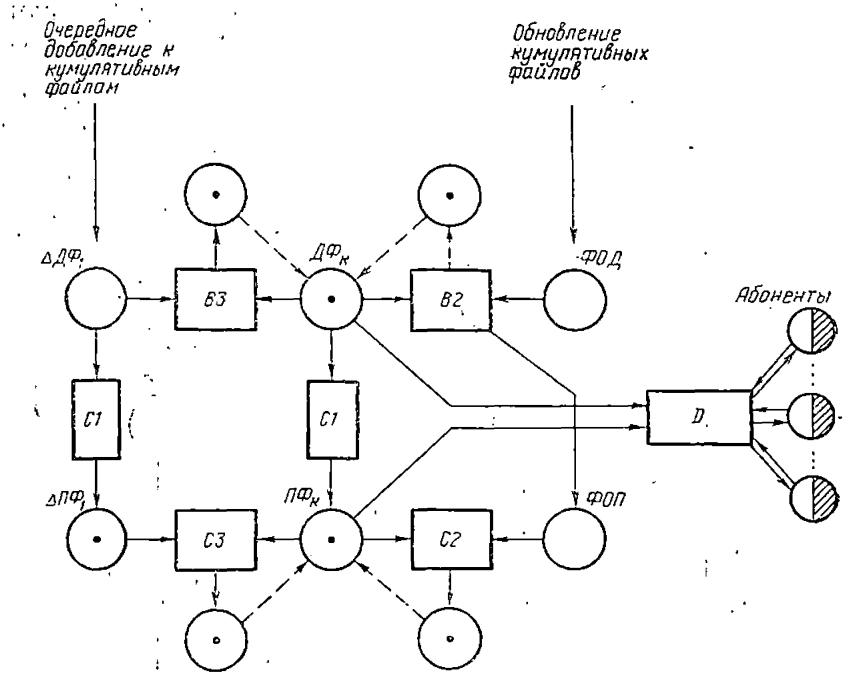



Рис. 6. Схема поддержки кумулятивных файлов $ДФ_к$ и $ПФ_к$ и проведения ретроспективного поиска по разовым запросам абонентов

и $ПФ_к$. Разовые запросы могут вводиться с перфокарт или пишущей машинки. Для обработки запросов в диалоговом режиме пользователь может создать собственные программы пред- и постобработки и использовать программу *D* в качестве подпрограммы.

В РПИ могут использоваться и архивы запросов, например для подготовки тематических библиографий по всему фонду документов.

*А. В. ЗАМУЛИН, Б. Н. ПИЩИК*

## ОБЕСПЕЧЕНИЕ В АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ НЕПРОЦЕДУРНЫХ ВОЗМОЖНОСТЕЙ РАБОТЫ С БАЗАМИ ДАННЫХ

Универсальная СУБД должна обеспечивать связь со следующими пользователями: администратором базы данных, системным программистом, прикладным программистом, непрограммис-

том, параметристом [1]. В данной статье рассмотрены возможности, предоставляемые пользователям двух последних типов, т. е. возможность спецификации простых запросов на поиск или модификацию информации в базе данных и возможность не задавать в запросе ничего, кроме определенных параметров заранее запрограммированных процедур обработки данных. В традиционных СУБД набор этих процедур обычно обеспечивается разработчиками системы, а его расширение выполняют системные программисты на языке разработки системы. Такова же ситуация и при необходимости расширения состава операторов, предназначенных для непрограммистов. Системные программисты при этом должны хорошо знать внутреннюю организацию системы. Этот факт обусловливает известную трудность развития таких систем.

Одним из способов построения объединенной СУБД, как указывалось в работах [2, 3], является ее реализация на основе алгоритмического языка, включающего типичные возможности автономных СУБД, либо способного порождать такие возможности. В соответствии со вторым подходом был разработан язык программирования Бояз, ориентированный на работу с базами данных [4].

Принципы Бояза существенно основываются на концепции, что СУБД не обязана иметь заранее развитые средства общения с непрограммистами и параметристами, а должна предоставлять аппарат для порождения и развития таких средств. Суммируя потребности пользователей-непрограммистов, к этому аппарату можно предъявить следующие требования:

возможность конструирования типов данных, адекватно отражающих формы пользовательских документов и взаимоотношений между ними;

возможность изображения конкретных документов наиболее естественным образом;

возможность образования нужного набора операторов директивного характера и его использования наиболее простым способом.

Таким образом, данные принципы предполагают, что, используя язык типа Бояза и построенную на его основе порождающую систему, системные программисты, входящие в штат администратора базы данных, развертывают конкретные системы в соответствии с потребностями того или иного приложения. Каждая конкретная система при этом будет обладать как процедурными возможностями исходного языка, так и непроцедурными возможностями порожденного языка.

В настоящей статье на примере языка Бояз иллюстрируются способы получения в алгоритмическом языке перечисленных выше возможностей непроцедурного характера, свойственных традиционным ИПС и автономным СУБД, ориентированным на сопряжение с непрограммистом.

**Типы данных.** База данных автономной СУБД представляет собой обычно набор связанных друг с другом файлов. Файл в свою очередь представляет собой последовательность записей одного

и того же или разных типов. Каждая запись является иерархической структурой постоянной, переменной или неопределенной длины. Переменный формат записи объясняется наличием в ней векторов (массивов, строк) и повторяющихся групп. Для обеспечения связей между файлами некоторые поля могут быть ссылками на записи того же самого или других файлов.

Во многих случаях основная информация базы данных дополняется вспомогательной в виде различных словарей, вторичных индексов и т. п. Это необходимо, с одной стороны, для обеспечения эффективного выполнения наиболее часто подаваемых запросов и, с другой стороны, для завершения формирования запроса с целью наиболее полного сканирования файлов базы данных (последней цели служат, например, тезаурусы информационно-поисковых систем).

Программные компоненты — процедуры, функции, макросы и т. п. — обычно не рассматриваются как элементы базы данных и выступают как часть программного обеспечения системы и хранятся в специальных библиотеках. Так как эти компоненты составляются на языке реализации системы, всякая их ревизия требует либо привлечения разработчиков системы, либо досконального изучения ее внутренней структуры.

На Боязе база данных может быть определена как совокупность переменных и констант следующих типов: целые и вещественные числа, литеры и строки из них, массивы, наборы, множества, записи и файлы, указатели, макросы, функции и процедуры. Определение типов данных рекурсивно, так что данные сложных типов (массивы, наборы, записи и файлы) могут содержать в качестве своих компонент данные любых типов. Такой подход к определению данных обеспечивает, с одной стороны, общность и, с другой стороны, гибкость в развертывании конкретных информационных систем или банков данных.

Существенной особенностью Бояза является то, что на уровень данных вынесены описания процедур и функций. Это дает возможность хранить их в базе данных наряду с данными других типов либо в виде самостоятельных переменных или констант, либо в виде компонент сложных типов данных (например, можно организовать файлы из процедур). Кроме того, средствами самого языка можно редактировать состав процедур, приспосабливая его к текущим потребностям приложения.

Язык не предполагает автоматического образования индексов, словарей и тому подобной вспомогательной информации. Вместо этого администратору базы данных (и соответственно входящим в его штат системным программистам) предлагается аппарат для организации необходимых структур. Занесение информации в эти структуры, а также их использование при поиске данных осуществляется сопровождающими их процедурами, хранящимися в базе данных.

В качестве примера предположим, что на основе базы данных «библиотека» необходимо развернуть информационно-поисковую

систему документального характера. Предположим также, что для поиска по ключевым словам необходимо создать тезаурус, в котором отображались бы (и автоматически использовались) отношения синонимии и родо-видовые связи каждого ключевого слова. В таком случае в определении нашей базы данных могли бы иметься следующие описания:

```
тезаурус: индекс-послед файл из
          запись
              термин: строк;
              род: ссылка на тезаурус;
              вид: набор из ссылка на тезаурус;
              син: набор из ссылка на тезаурус;
          конец ключ термин
фонд: послед файл из
              запись
          название: строк;
          автор: набор из строк;
          ключ: набор из ссылка на тезаурус;
          инвномер: цел
              конец
```

В приведенных примерах встречаются различные типы данных. Описатель **строк** означает, что значением соответствующего элемента данных будет некоторая строка символов. Описатель **ссылка на** вводит тип указателя, т. е. соответствующий элемент данных будет содержать ссылку на компоненту указанного файла. Описатель **набор** сообщает, что элемент данных представляет собой набор однотипных компонент, в частности — набор ссылок на различные компоненты одного и того же файла. Соответственно описатель **запись** вводит новый тип записи, а описатель **файл** — тип файла.

Получение или изменение значения любой переменной осуществляется ее употреблением соответственно в левой или правой части оператора присваивания. Обмен данными с файлом производится через буфер файла, который обозначается переменной-файл, сопровождаемой ромбиком (т. е. для файла с именем Ф буфер файла будет иметь обозначение Ф ◊), и имеет тип компоненты файла. Для пересылки данных между буфером и файлом существуют соответствующие процедуры.

Данные любых типов (кроме файлов) могут быть представлены посредством изображений. Значения целого и вещественного типов изображаются соответственно целыми и вещественными числами, значения литерного типа — литерами, заключенными в кавычки, а строки — последовательностью литер в кавычках. Массивы, наборы и записи изображаются перечислением в квадратных скобках своих компонент. Специальный символ **nil** обозначает отсутствие значения у данных любого типа. Конкретные примеры изображений и описание процедур работы с базой данных примера будут приведены в последующих разделах.

**Описание данных.** Описание данных обычно является прерогативой администратора базы данных и заключается в составлении схемы базы данных, в соответствии с которой осуществляется ло-

гическая организация и выборка данных базы данных. В Боязе имеется возможность как составлять исходную схему базы данных, так и расширять и сокращать ее в течение всего времени существования базы данных.

Исходная схема создается описанием базы данных, которое имеет вид:

**база** имя-базы-данных
совокупность-описаний **конец**

В «совокупности описаний» администратор определяет типы данных и описывает переменные и константы, свойственные создаваемой базе данных. Рассматривая пример предыдущего раздела, можно сказать, что для базы данных «библиотека» совокупность описаний заключает в себе описание переменных «тезаурус» и «фонд». Для каждой из определяемых переменных (в том числе и для самóй базы данных) администратор может специфицировать замки защиты на доступ и модификацию, определяя тем самым круг и полномочия пользователей, взаимодействующих с описываемой базой данных. Если в дальнейшем появится необходимость расширить базы данных введением какой-либо новой переменной или константы, достаточно поместить в какой-либо программе соответствующее описание объекта с пометкой **глоб**. После выполнения этой программы новое описание будет доступно всем следующим по времени выполнения программам (разумеется, в рамках указанной базы данных). Некоторые автономные системы (например, СОКРАТ [5]) позволяют описывать условные элементы данных, существование которых в каждом экземпляре описываемого объекта зависит от существования или значения других элементов данных. Этот аспект описания, на наш взгляд, больше относится к контролю данных и поэтому будет рассмотрен в соответствующем разделе.

**Структура программы.** Первым требованием к программе со стороны непрофессиональных программистов является ее простота. В простейшем случае в программе пользователя должны быть указаны лишь имя базы данных, с которой он собирается работать, и директива на выполнение конкретного действия. В соответствии с этим требованием программа на языке Бояз имеет следующий вид:

[**пароль** список паролей пользователей]
оператор.

Здесь квадратные скобки обозначают факультатив, т. е. пользователь, работающий с незащищенными частями базы данных, может не предъявлять паролей. Оператор, составляющий тело программы, обычно будет иметь вид оператора над базой данных:

**над** имя базы данных **делай** оператор1.

Посредством имени базы данных пользователь идентифицирует базу, с которой он желает работать, а оператор 1 задает конкретные действия, которые необходимо совершить. Для программиста

последний будет, скорее всего, составным оператором, имеющим вид:

**начало** последовательность описаний и операторов **конец**

и содержащим, по сути дела, программу обработки или модификации указанной базы данных. Для непрограммиста и параметриста это будет оператор сканирования файлов или оператор обращения к глобальной процедуре, как показано в следующих разделах.

**Ввод данных.** Ввод новых данных в автономной СУБД осуществляется обычно подачей соответствующей директивы, за которой следует комплект вводимых данных, записанных в соответствии с соглашениями синтаксиса входного языка (так действует в пакетном режиме команда «С» системы СОКРАТ [5], оператор создания документов системы ВЕГА [6] и т. д.). Данные, прошедшие синтаксический и семантический контроль, записываются в соответствующие файлы базы данных, а ошибочные данные либо отвергаются совсем, либо помещаются в один из вспомогательных файлов, где они впоследствии могут быть скорректированы операторами редактирования (так, например, сделано в системе ИНФОР [7]). Источником данных в общем случае может быть любой носитель (перфокарты, магнитная лента, диск и т. д.). При наличии возможностей диалогового ввода система обычно печатает имя очередного вводимого элемента данных, а пользователь сообщает ей вводимое значение.

В Боязе ввод организуется самой программой пользователя или вызываемыми ею глобальными процедурами (см. приложение). Вследствие этого ввод данных может быть осуществлен как в диалоговом, так и в пакетном режиме со всех носителей, имеющихся в комплекте вычислительной машины. Для облегчения организации ввода-вывода язык имеет в своем составе такую конструкцию, как «поток», который отображает одно из внешних устройств ЭВМ и предназначен для обмена данными между базой данных и внешней средой.

В простейшем случае каждая СУБД предлагает своим пользователям стандартный формат для оформления вводимых данных. Например, в системе СОКРАТ значения элементов данных отделяются друг от друга знаком «/», в системе ВЕГА используются для этого запятые и точка с запятой. Точно таким же образом Бояз предлагает стандартный вариант оформления вводимых данных, при котором значение каждой переменной или константы представляется своим изображением. Например, одна из записей файла «фонд» могла бы иметь следующее изображение:

```
['система управления базами данных';
 ['А. П. Иванов', 'Г. Д. Сидоров'];
 ['СУБД', 'база данных', 'банк данных'];
17453]
```

Для ввода таких записей из потока «мойввод» в файл «фонд» можно написать следующую процедуру:

```
проц вводфонд = начало повтор ввод(мойввод,фонд◇);
                              put (фонд);
```

```
        до конец-потока(мойввод);
конец
```

Если пользователь не хочет вводить записи стандартным способом, он может вводить их покомпонентно, помещая между полями удобные для него разделители; приведенная выше процедура «вводфонд» в таком случае выглядела бы несколько сложнее. Составленная таким образом процедура может быть объявлена глобальной процедурой и в последующих программах использоваться как директива; например, программа ввода документов в файл «фонд» могла бы выглядеть следующим образом:

```
над библиотека делай вводфонд
```

Сложнее обстоит дело с комплектованием тезауруса ввиду того, что для каждого вводимого в него термина необходимо устраивать перекрестные ссылки между синонимами и терминами, находящимися в родо-видовых отношениях. В приложении приводится пример процедуры «втезар», которая помещает новый термин в тезаурус и отмечает все его связи. Эта процедура имеет тип:

```
проц (строк, строк, набор из строк, набор из строк)
```

и может быть использована таким же образом, как и процедура «вводфонд». Например, при необходимости ввода в тезаурус термина «мебель», имеющего родовой термин «деревянные изделия», видовые термины «стол» и «стул» и не имеющего синонимов, можно составить следующую программу:

```
над библиотека делай втезар ('мебель', 'деревянные изделия',
                             ['стол', 'стул'], nil)
```

Заметим, что в конкретной системе могли быть написаны как более общие, так и более частные процедуры ведения тезаурусов (например, ввод группы терминов или установление конкретных связей между существующими терминами).

**Поиск данных.** В универсальной СУБД пользователь должен иметь возможность задавать системе разнообразные вопросы. Это означает, что он может специфицировать выборку записей по любой комбинации их полей с наложением различных критериев выборки на каждое поле и осуществить выделение любого множества полей из записей, удовлетворяющих специфицированным критериям. При этом задание пользователя должно представляться в максимально упрощенном виде.

Как отмечалось в работе [8], независимо от архитектуры входного языка в поисковых заданиях автономных СУБД всегда можно различить две части: спецификационную и операционную. Первая часть указывает системе, какой набор данных необходимо выделить в базе данных, а вторая — что необходимо сделать с выделенными данными. Очевидно, что язык программирования, ориентированный на работу с базами данных, должен предоставлять непрофессиональным программистам аналогичные возможности. В Боязе для этой цели существует оператор сканирующего цикла:

в переменная-файл [для выражение] делай оператор.

Здесь выражение является логическим выражением над полями записи и задает критерий выборки записей в файле, представленном переменной-файл. Этот критерий может быть как элементарным, так и более сложным.

Элементарные критерии выборки служат для проверки значения поля записи. Это осуществляется посредством шести стандартных операций отношения $=$, $\neq$, $>$, $<$, $\geqslant$, $\leqslant$ (операнды операции имеют один и тот же тип) и шести операций включения **сод, содб, содм, ⌉сод, ⌉содб, ⌉содм** (левый операнд — набор или множество, правый операнд — компонента соответственно набора или множества). Более сложные критерии строятся из элементарных посредством логических операций $\wedge$ (и), $\vee$ (или), $\neg$ (не), вовлекающих в рассмотрение значения нескольких полей записи. Следующий этап — введение так называемых макрофункций, благодаря которым пользователь получает гибкое средство для порождения удобного языка критериев.

Набор подходящих макрофункций определяется при развертывании конкретного приложения СУБД. Поэтому в настоящей статье не приводится каких-либо новых критериев, вводимых с помощью макросредств, а лишь показывается, что с их помощью легко могут быть описаны критерии выборки некоторых известных систем (квантификаторы и «порядковые» местоимения системы СОКРАТ, проверочные функции *top*(*N*, *X*, *D*) и *boffom*(*N*, *X*, *D*) подъязыка АЛЬФА [9] реляционной СУБД и т. п.). Описав макрос

*mytop* (имя файла: **строк**, имя поля: **строк**) **строк**,

вырабатывающий тело функции, выбирающей в файле «имя файла» запись с максимальным значением поля «имя поля», мы сможем употребить его в качестве одного из параметров процедуры «печать»; например:

печать (*mytop* (фонд, инвномер), название)

распечатает название документа, имеющего наибольший инвентарный номер.

В качестве второго примера опишем макрос, реализующий квантификатор **каждый** системы СОКРАТ:

```
макро кажд=(т : строк) строк
    начало м : строк, а : лит, i, j : цел, тек : строк ;
      i : = 1, тек : = "начало p : = ложь; в";
   пока т[i]≠'('делай начало м[i] : = т[i]; i : = i+1 конец;
тек : = тек плюс м;
      тек : = тек плюс 'делай если ⌉ ';
      м : = nil; j : = 0; i : = i — 1 ;
     повтор i : = i+1; j : = j+1; м[j] : = т[i] до т[i] = ')';
     тек : = тек плюс м;
     тек : = тек плюс 'то на вон; p : = истина;
                              вон : рез p конец';
    рез тек
конец
```

Пусть теперь машина есть набор, состоящий из записей с полями 'марка' и 'цвет'. Тогда обращение к макрофункции

кажд (машина (марка = 'а' ∧ цвет = 'в'))

будет переведено в тело функции:

```
начало р : = ложь ;
   в машина делай если ¬ (марка = 'а' ∧ цвет = 'в')
                  то на вон;
   р : = истина ;
   вон : рез р
конец
```

принимающей значение **истина** только тогда, когда у всех компонент набора поля «марка» и «цвет» будут иметь значения соответственно 'а' и 'в'.

Продолжая рассмотрение сканирующего цикла, отметим, что оператор задает алгоритмические действия, которые необходимо произвести над выделенными записями. В частности, здесь можно запросить вывод на АЦПУ (или другое воспроизводящее устройство) либо всей записи целиком, либо любой совокупности ее полей. Например, для поиска в файле «фонд» может быть подан следующий оператор:

```
в фонд для ключ сод ['анализ', 'система файлов']
   делай печать (название, автор, инвномер),
```

где «печать» может быть макрофункцией или глобальной процедурой, распечатывающей в определенном формате значения указанных полей записи.

**Обновление данных.** Характерным свойством функционирующей СУБД является динамическое состояние ее информационных массивов. В базу данных вводится новая информация, удаляется или изменяется старая. При этом обновление данных может производиться как на уровне записей, так и на уровне элементов данных (полей записи). Если же допускается обновление на уровне отдельного символа, система управления базами данных начинает дополнительно выполнять функции текстового редактора.

В автономных системах обычно имеется фиксированный набор директив для модификации информации (**удалить, изменить, ввести** — в системе ВЕГА [6], А, М, Е, — в системе СОКРАТ [5], ЗАМЕНИТЬ, СЛИТЬ, УДАЛИТЬ — в системе ИНФОР [7]). Эти директивы изобретаются разработчиками системы, и их состав и формат выбирается таким, чтобы покрыть потребности предполагаемого круга приложений системы. Как и в случае с поиском, появление новой функции обновления ведет к необходимости расширения исходного состава директив.

В Боязе, как и в других алгоритмических языках, основным средством изменения значения переменной является оператор присваивания. Это положение не выполняется лишь для файлов и наборов: поскольку они представляют собой образцы «резиновых» переменных, для изменения их значения, т. е. для помещения в них

новых компонент и удаления старых, служат специальные процедуры («пом» и «удал» для наборов, *put* и *eraze* для файлов).

Обновление данных посредством оператора присваивания будет, по-видимому, более естественным для пользователей-программистов. Непрограммисты, очевидно, пожелают, чтобы обновление сохраняло директивный характер. Для этой цели может использоваться уже упоминавшийся аппарат глобальных процедур и макрофункций, а выделение данных, подлежащих модификации, может попрежнему осуществляться в рамках сканирующего оператора. Фильтрующее выражение этого оператора позволяет выделять в сканируемом цикле требуемое множество записей (от одной до всего файла), благодаря чему обновлению может подвергнуться как отдельная запись, так и определенная совокупность записей.

Для примера вернемся к рассмотрению файла «тезаурус». Естественно предположить, что по мере накопления документов в тезаурусе будут появляться как новые термины, так и новые связи между ними. Тогда для установления между существующими терминами родо-видовых связей и отношений синонимии могли бы быть написаны следующие глобальные процедуры: тезрод (терм: **строк**), тезвид (терм: **набор из строк**) и тезсин (терм: **набор из строк**). В качестве примера использования этих процедур можно привести следующий оператор сканирующего цикла:

```
в тезаурус для супервизор делай
        начало тезрод ('операционная система');
            тезвид ('планировщик');
            тезсин ('диспетчер', 'монитор')
        конец
```

Характерной особенностью этого оператора является то, что он позволяет производить пакетирование как обновляющих, так и поисковых операторов. В данном примере объединены в пакет три обновляющих оператора таким образом, что термин «супервизор» связывается родовой связью с «операционной системой», видовой связью — с «планировщиком» и отношением синонимии — с «диспетчером» и «монитором».

Дополнительным преимуществом Бояза в отношении обновления является то, что строковое значение рассматривается в нем как набор символов. Благодаря этому обновление данных можно доводить до уровня отдельного символа (вычеркивание, вставка, замена символов) с выполнением тем самым функций редактора текста.

**Контроль данных.** Одним из важных требований, предъявляемых к СУБД, является обеспечение корректности информации, внесенной в базу данных, и уменьшение числа модификаций, связанных с ошибками при вводе. В известных системах существуют различные методы семантического контроля вводимых данных. В системе ВЕГА — это динамический набор условий, которым должны удовлетворять вновь создаваемые документы, в системе СОКРАТ — это условные элементы данных, в системе ИНФОР — определенный формат ввода в массивы для редактирования и т. д. Од-

нако во всех случаях имеют место два основных критерия контроля: критерий существования и критерий значения. Критерий существования означает, что элемент данных должен (или не должен) существовать во вводимом экземпляре записи безусловно или в зависимости от значений других элементов данных, а критерий значения проверяет отношение значения элемента данных к специфицированной пользователем величине (также условно или безусловно).

Применение указанных выше критериев, очевидно, сводится к вычислению соответствующих логических функций. Таким образом, перед занесением информации в базу данных могут быть описаны отдельные глобальные функции или даже наборы (файлы) из таких функций контроля, которые затем связываются с входными потоками и файлами данных (основным хранилищем информации). Именно такая идеология проводится в Боязе. Пусть имеется файл проверочных функций ПФ для контроля данных, поступающих в файл «фонд». Тогда вместо ранее описанной процедуры «ввод-фонд», осуществляющей бесконтрольный ввод данных, можно описать процедуру ввода с контролем:

```
проц провводфонд =
начало повтор ввод(мойввод, фонд ◇ );
        в ПФ делай
              если ¬ ПФ ◇
              то начало печать(ошибка) ; на вон;
                    конец
              put  (фонд);
   вон: до конец-потока(мойввод)
конец
```

Файл контрольных функций обновляется таким же образом, как и любой другой файл, благодаря чему состав контрольных функций может динамически изменяться без особых усилий (что, например, трудно сделать в СОКРАТе) и без привлечения специального аппарата (как было, например, в ВЕГЕ).

**Использование файлов из функций и процедур.** Файлы из процедур и функций вообще представляют собой полезный аппарат для реализации различных свойств СУБД. В качестве еще одного примера возьмем системы избирательного распределения информации. Как известно, в таких системах постоянно хранятся запросы различных пользователей, которые периодически прокручиваются на файлах новых поступлений литературы. Очевидно, что каждый запрос можно представить как процедуру, осуществляющую поиск информации в определенном файле и выводящую ее на определенное устройство в определенном формате. Совокупность запросов одного пользователя можно представить тогда как файл из таких процедур с вытекающими отсюда возможностями динамического обновления этой совокупности. Выполнение таких запросов можно изобразить следующим образом.

Предположим, что индивидуальные совокупности запросов описываются следующими файлами:

```
мойзапрос : файл из проц
егозапрос : файл из проц
. . .
```

Выполнение процедур каждого из этих файлов при поиске информации в файле «фонд» может быть в свою очередь описано следующими процедурами второго уровня:

```
проц мояпроц = начало в фонд делай мойзапрос конец
проц егопроц = начало в фонд делай егозапрос конец
. . .
```

Следующий уровень — файл процедур избирательного распределения информации:

```
файл ири : файл из проц
```

После выполнения последовательности операторов:

```
ири ◇ : = мояпроц; put (ири);
ири ◇ : = егопроц; put (ири);
. . .
```

файл «ири» будет содержать все процедуры второго уровня. Теперь оказывается, что программа, содержащая единственный оператор

```
ири;
```

будет выполнять всю работу по избирательному распределению информации.

Приведенные рассуждения и примеры показывают, что непроцедурные возможности могут быть созданы как расширения входного языка СУБД, являющегося по своей природе алгоритмическим языком и ориентированного на работу с базами данных. Из тех же примеров ясно, что система, спроектированная на основе такого языка, не является (по крайней мере для непрограммистов) готовой системой, а представляет собой скорее полуфабрикат, который нужно подогнать под потребности конкретного приложения. Поскольку допрограммирование системы осуществляется на ее входном языке, такая подгонка не вызывает затруднений. В то же время сохраняется универсальность системы и возможность достаточно легкого отслеживания меняющихся потребностей приложения.

В рассматриваемом языке составление глобальных процедур (процедур базы данных) ведется в соответствии с канонами традиционных языков программирования. Дальнейшее усовершенствование этого процесса, очевидно, должно производиться с учетом работ по абстрактным типам данных [10] и родовым процедурам [11].

## *ПРИЛОЖЕНИЕ*

Процедура ввода термина в тезаурус

```
проц втезар = (терм : строк , прод : строк ,
               набвид : набор из строк ,
               набсин : набор из строк)
    начало перем уктерм : ссылка на тезаурус,
                укрод : ссылка тезаурус,
                уквид : набор из ссылка на тезаурус,
                уксин : набор из ссылка на тезаурус,
                i : цел;
           дек (тезаурус, терм);
           если стрелка (тезаурус) ≠ nil то на вок;
               {для существующих в тезаурусе терминов.
                процедура не работает}
           тезаурус ◇ : = nil
           тезаурус ◇ . термин : = терм;
           put (тезаурус);
           уктерм : = стрелка (тезаурус);
               {записали термин в тезаурус и запомнили
                ссылку на него}
           если прод = nil то на виды;
               {если не указан родовой термин, переходим на
                обработку видовых терминов}
                get (тезаурус, прод);
                если стрелка (тезаурус) = nil
                то начало тезаурус ◇ : = nil
                          тезаурус ◇ . термин : = прод;
                   конец
                пом (тезаурус ◇ . вид, уктерм);
            put (тезаурус);
           укрод : = стрелка (тезаурус);
           {если родового термина не было в тезаурусе,
           формируем его в буфере файла, затем в любом слу-
           чае дополняем термин ссылкой на видовой термин,
           отправляем его в тезаурус и запоминаем ссылку на
           него}
    виды: если набвид = nil то на сины;
              {если не указаны видовые термины, переходим на обработку си-
              нонимов}
          для i : = 1 до текдл (набвид) делай
               начало дек (тезаурус, набвид [i]);
                      если стрелка (тезаурус) = nil
                      то начало тезаурус ◇ : = nil;
                           тезаурус ◇. термин : = набвид [i];
                       конец;
```

```
                    тезаурус ◇. род : = уктерм;
                    put (тезаурус);
                    пом (уквид, стрелка (тезаурус));
                конец
                {в цикле для каждого термина проверяем его наличие в сло-
                 варе, в случае необходимости формируем новый термин, до-
                 полняем термин ссылкой на родовой термин, отправляем его
                 в тезаурус и запоминаем ссылку на него}
сины : если набсин = nil то на затерм;
          {если не указаны синонимы, переходим на заполнение термина}
       для i : = 1 до текдл(набсин) делай
            начало дек(тезаурус, набсин [i]);
                   если стрелка(тезаурус) = nil
                  то начало тезаурус : = nil;
                       тезаурус◇. термин : = набсин [i];
                      конец
                   пом(тезаурус ◇. син, уктерм);
                    put (тезаурус);
                    пом(уксин, стрелка(тезаурус));
            конец
затерм: дер (тезаурус,уктерм);
  с тезаурус ◇ делай
        начало род : = укрод;
               вид : = уквид:
               син : = уксин;
        конец {заполнили введенный термин ссылками на родовой, видовой и
               синонимичные термины};
                        put (тезаурус);
вон: конец {процедуры «втезар»}
```

## ЛИТЕРАТУРА


1. Столяров Г. К., Дрибас В. П. Основные требования к банкам данных. — УСиМ, 1975, № 2, с. 40—48.

2. Olle T. W. Current and future trends in data — base management systems: «Information processing 74». Procc: IFIP Congr., 1974, book 5, p. 998—1006.

3. Замулин А. В. О языке программирования, ориентированном на работу с базами данных. — Программирование, 1975, № 5, с. 23—31.

4. Замулин А. В. Бояз — язык программирования, ориентированный на работу с базами данных. — В кн.: Алгоритмы и организация решения экономических задач. М., Статистика, 1978, вып. 12.

5. Abrial G. R. et al Project SOCRATE. Specifications generales. Universite de Grenoble. Institut de Mathematiques Appliquées. Grenoble, août, 1970.

6. Замулин А. В., Богданова Г. С., Бородин О. В. и др. Архитектура информационно-поисковой системы ВЕГА. — УСиМ, 1975, № 6, с. 17—24.

7. Андон Ф. И., Стогний А. А. Система ИНФОР. Общее описание. — УСиМ, 1974, № 4, с. 26—31.

8. Замулин А. В. Языковые вопросы построения информационно-поисковых систем. — В кн.: Вопросы обучения языкам программирования. Киев, 1974, с. 62—71.

9. Codd E. F. A Data base sublanguage Founded on the relational Calculus. — Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control.

10. Liskov B. and Zilles S. Programming wiht abstract data types. — Proc. of ACM SIGPLAN Conference on Very High Level Languages. SIGPLAN-Notices, 9(April, 1974), 50—54.

11. Gries D., Gehani N. Some Ideas on Data Types in High — Level Languages. Comm. ACM 20, 6 (June 1977), p. 414—420.

## АННОТАЦИИ СТАТЕЙ СБОРНИКА

Мясников В. А. **Задачи и перспективы развития, производства и использования вычислительной техники в народном хозяйстве страны.**

Раскрываются основные проблемы производства, развития и применения вычислительной техники в народном хозяйстве. Даются общие характеристики и текущее состояние ОГАС, ОГСПД, ГСВЦ, а также перспективы их развития.

Савинков В. М., Казаров М. С., Рысевич Ю. К. **Использование систем управления базами данных в АСУ.**

Дается обзор состояния работ по СУБД и приводятся основные характеристики отечественных пакетов прикладных программ, обеспечивающих создание баз данных в АСУ.

Каменцев А. А. **Организация структуры данных при регистровой форме хранения и обработки информации.**

Описана структура регистровой формы наблюдения за объектами статистической отчетности.

Мишенин А. И. **Параметры, влияющие на выбор структуры данных.**

Анализируются методы доступа к последовательным и другим структурам данных, размещенных в оперативной памяти. Предлагается А-метод организации структур.

Гарусов В. Н., Голубков В. Г., Лорман В. А. **ППП для решения задач линейного программирования итеративными методами с применением диалогового аппарата.**

Статья посвящена описанию ППП «СОЛМИ-ЕС», который предназначен для автоматизации формирования матриц задач линейного программирования и решения их итеративными методами. Пакет реализован на базе ОС ЕС.

Васильев П. П., Евсеев Ю. В. **Производительность труда программистов.**

Приводятся зависимости продолжительности разработки программы от ее размера и квалификации программиста. Предлагается способ прогнозирования сроков разработки программ на ЭВМ.

Иванов Н. О., Макаренко Е. И., Сурнов С. И. **Конструктор управляющих программ.**

Описывается алгоритм конструирования входного языка и управляющей программы пакета прикладных программ.

Меркушов Ю. П. **Анализ некоторых особенностей выдачи диагностических сообщений Ассемблером ОС ЕС ЭВМ.**

Дается описание особенностей выдачи диагностических сообщений для нестандартных ситуаций, которые возникают при анализе синтаксических ошибок в программах, написанных на Ассемблере.

Алексеенко Е. А., Довгялло А. М., Небрат О. П., Платонов Б. А., Стогний А. А. **Система программирования и поддержания обслуживающих и обучающих курсов.**

Система СПОК обеспечивает создание на ЕС ЭВМ обслуживающей и обучающей среды, повышающей эффективность подготовки пользователей ЭВМ и улучшающей условия их работы.

Фридлендер Ф. Л., Гордонова В. И. **Подсхема для языка ПЛ/1 в системе управления базами данных сетевой структуры.**

Изложена концепция определения данных подсхемы для языка ПЛ/1, развивающая предложения рабочей группы DBTG системного комитета CODASYL и используемая при разработке ППП СУБД.

Столяров Г. К. **Языки и функциональные возможности системы АСПИД-3.**

Рассматриваются состав и функциональные возможности программ пакета АСПИД-3, работающего на ЕС ЭВМ под управлением ДОС и предназначенного для организации документальной информационно-поисковой системы дескрипторного типа с нормированной лексикой и инвертированным поисковым файлом.

Замулин А. В., Пищик Б. Н. **Обеспечение в алгоритмическом языке непроцедурных возможностей работы с базами данных.**

На примере системы Бояз иллюстрируются возможности порождения в алгоритмическом языке непроцедурных средств для пользователей-непрограммистов, свойственных традиционным ИПС и автономным СУБД.

## СОДЕРЖАНИЕ

А45 **Алгоритмы** и организация решения экономических задач: Сб. статей/Редкол.: В. М. Савинков (пред.) и др. — Вып. 13. — М.: Статистика, 1979. — 174 с., ил.
65 к.

Статьи очередного выпуска сборника освещают важные проблемы создания общегосударственной автоматизированной системы, знакомят с новыми пакетами прикладных программ, обеспечивающих работу с базами данных, информационно-поисковых систем, обучение пользователей с помощью ЭВМ.

Сборник рассчитан на руководящих и научно-технических работников, программистов и проектировщиков автоматизированных систем.

А $\frac{30502-034}{008(01)-79}$ 72—79 1502000000

ББК 32.973
6Ф7.3

**АЛГОРИТМЫ И ОРГАНИЗАЦИЯ РЕШЕНИЯ ЭКОНОМИЧЕСКИХ ЗАДАЧ**

*Вып. 13*

Редактор *Т. А. Петрова*

Мл. редактор *Г. В. Розанова*

Техн. редактор *И. В. Завгородняя*

Корректоры *Г. В. Хлопцева, Г. И. Терновская*

Худ. ред. *Э. А. Смирнов*

ИБ № 697

---

Сдано в набор 16.10.78. Подписано в печать 08.02.79.
А08545. Формат 60×90¹/₁₆. Бум. тип. № 1. Гарнитура «Литературная».
Печать высокая. П. л. 11. Усл. п. л. 11. Уч.-изд. л. 11,95. Тираж 16 000 экз.
Заказ 2407. Цена 65 коп.

---

Издательство «Статистика», Москва, ул. Кирова, 39
Великолукская городская типография управления издательств,
полиграфии и книжной торговли Псковского облисполкома,
г. Великие Луки, ул. Полиграфистов, 78/12