Э.Н. Хотяшов

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ АСУ

Э. Н. ХОТЯШОВ

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ АСУ

ИЗДАТЕЛЬСТВО «ВЫШЭЙШАЯ ШКОЛА»
МИНСК 1974

6П2.15
Х 85
УДК 681.3 : 65

Гос. публичная
научно-техническая
библиотека
ЭЛЕМ
Ж-34211

Хотяшов Э. Н.

Х 85 Математическое обеспечение АСУ. Минск, «Вышэйш. школа», 1974.

352 с. с ил.

В пособии приведена классификация математического обеспечения автоматизированных систем управления предприятием (МО АСУП). Определено понятие системы программирования для АСУП. Изложены структура операционной системы применительно к ЭВМ «Минск-32» и описание алгоритмического языка КОБОЛ, рассмотрены процедуры обработки экономической информации, дан обзор методов решения экстремальных задач. На примере одной из задач АСУП рассмотрена характеристика этапов проектирования системы обработки данных.

Пособие предназначено для специалистов вычислительных центров, проектных и научно-исследовательских институтов и организаций, занимающихся вопросами разработки математического обеспечения АСУП. Может быть использовано студентами высших учебных заведений, аспирантами и слушателями курсов переподготовки и повышения квалификации в качестве учебного пособия.

Х $\frac{0316-092}{М\ 304(05)-74}$ 53—74

6П2.15

© Издательство «Вышэйшая школа», 1974 г.

ПРЕДИСЛОВИЕ

В наше время широко развертывается работа по созданию и внедрению автоматизированных систем управления (АСУ) в народное хозяйство. Накоплен значительный опыт внедрения АСУ предприятиями. Так, функционируют системы управления на Минском и 2-м Московском часовых, Львовском телевизионном, Минском тракторном заводах, на заводе электронных вычислительных машин им. Г. К. Орджоникидзе, Павлодарском тракторном заводе и многих других.

Основой всех автоматизированных систем управления предприятиями является система электронной обработки данных (СЭОД). В свою очередь существенная доля затрат при создании СЭОД падает на разработку математического обеспечения. Анализ распределения затрат на разработку целого ряда АСУП, которая проводилась в Центральном научно-исследовательском и проектно-технологическом институте организации и техники управления (ЦНИИТУ), показывает, что затраты на разработку математического обеспечения составляют примерно 35% от общих затрат. При этом наблюдается явная тенденция к увеличению этой доли.

В силу того что в настоящее время нет однозначного толкования, что же включается в математическое обеспечение (МО) АСУП, в главе 1 проводится классификация общей структуры МО АСУП.

Отметим, что в данной книге основное внимание уделено вопросам разработки математического обеспечения АСУП, но в силу общности ряда вопросов они будут приемлемы и при разработке отраслевых автоматизированных систем управления, систем управления строительством и др.

Существенным является понятие системы программирования для АСУП. Под системой программирования для АСУП понимается некоторая система языковых и программных средств, обеспечивающая как эффективную возможность разработки программ для решения конкретных задач АСУП, так и создание проблемно-ориентированных систем программирования. В систе-

му программирования включены управляющая часть операционной системы, ассемблер, транслятор с языка КОБОЛ, система типовых процедур обработки данных, организованная в виде библиотеки стандартных программ и генераторов.

Важность составных частей системы программирования предопределила содержание и последовательность изложения 2-й, 3-й и 4-й глав книги. ЭВМ «Минск-32» нашла достаточно широкое применение для решения рассматриваемых проблем. Это явилось одной из основных причин того, что при изложении материала за основу была взята именно эта ЭВМ. В главе 2 рассматриваются характеристики операционной системы и системы символического кодирования ЭВМ «Минск-32», в главе 3 — библиотека типовых процедур обработки данных. Так как КОБОЛ занимает важное место в системе программирования для АСУП, а литературы с его описанием недостаточно, в главе 4 изложены основные положения входного языка КОБОЛ для ЭВМ «Минск-32». В качестве основы был использован вариант входного языка САОД [10]. При этом были уточнены некоторые конструкции входного языка, расширены примеры, уточнена терминология. Так, вместо «длинного» термина «элемент данных» автор счел целесообразным использовать достаточно распространенный термин «реквизит». В данной работе эти термины нужно понимать как эквивалентные по значению.

В главе 5 сделан обзор ряда методов решения экстремальных задач и приведены блок-схемы алгоритмов этих методов. В главе 6 рассмотрены этапы разработки математического обеспечения задачи из класса АСУП, изложение иллюстрируется достаточно сложными примерами некоторых задач из подсистемы реализации и сбыта продукции.

Книга предназначена для специалистов вычислительных центров, проектных и научно-исследовательских институтов и организаций, занимающихся вопросами разработки математического обеспечения АСУ. Книга может быть использована студентами высших учебных заведений, аспирантами и слушателями курсов переподготовки и повышения квалификации в качестве учебного пособия.

Глава 4 написана совместно с В. И. Беличенко, глава 5 — с В. И. Комликом, глава 6 — с В. В. Лазовиком.

При подготовке рукописи существенную помощь оказали А. А. Герман, В. Н. Агафонов, В. И. Липницкий, А. Н. Шеметов. Всем им автор выражает искреннюю признательность.

Особую благодарность автор выражает профессору В. В. Шкурбе, взявшему на себя нелегкий труд рецензента. Его замечания способствовали существенному улучшению книги.

Автор

Глава 1

ОБЩАЯ СТРУКТУРА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ (АСУП)

1.1. КЛАССИФИКАЦИЯ МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ АСУП

Разработка математического обеспечения является одной из основных работ при проектировании и внедрении АСУП.

Рассмотрим некоторую общую классификацию математического обеспечения АСУП. Разделим общее математическое обеспечение АСУП на шесть классов (рис. 1.1).

Первый класс — системы математического обеспечения ЭВМ, или внутреннее математическое обеспечение ЭВМ. Системы математического обеспечения ЭВМ (СМО ЭВМ) являются как бы развитием аппаратурных возможностей ЭВМ, но реализованных с помощью специальных программ. Обычно СМО ЭВМ создается организацией, которая разрабатывает технические средства соответствующей ЭВМ.

Второй класс — алгоритмы обработки данных. В зависимости от возможностей машины (объема оперативной памяти, наличия памяти на магнитных дисках, барабанах или лентах и их объема, быстродействия, количества и разнообразия периферийных устройств), характера решаемой задачи, объемов перерабатываемой информации структура информационных массивов будет различной, будут различны и алгоритмы переработки таких массивов.

Третий класс — некоторые методы решения экстремальных задач. Целый ряд задач АСУП требует применения экономико-математических методов оптимизации. К таким задачам относятся: формирование производственной программы предприятия, составление календарных графиков запуска-выпуска, управление запасами материалов и полуфабрикатов, управление реализацией и сбытом продукции и др.

Четвертый класс — система стандартизации элементов проектирования математического обеспечения АСУП. Разработка программ разбивается на ряд этапов, включающих составление алгоритма решения задачи, составление принципиальных и рабочих блок-схем алгоритма, запись программы на одном из языков программирования, отладку программы на ЭВМ, опробование программы на реальных данных, внедрение на конкретном объекте. При этом на разных этапах могут участвовать специалисты раз-

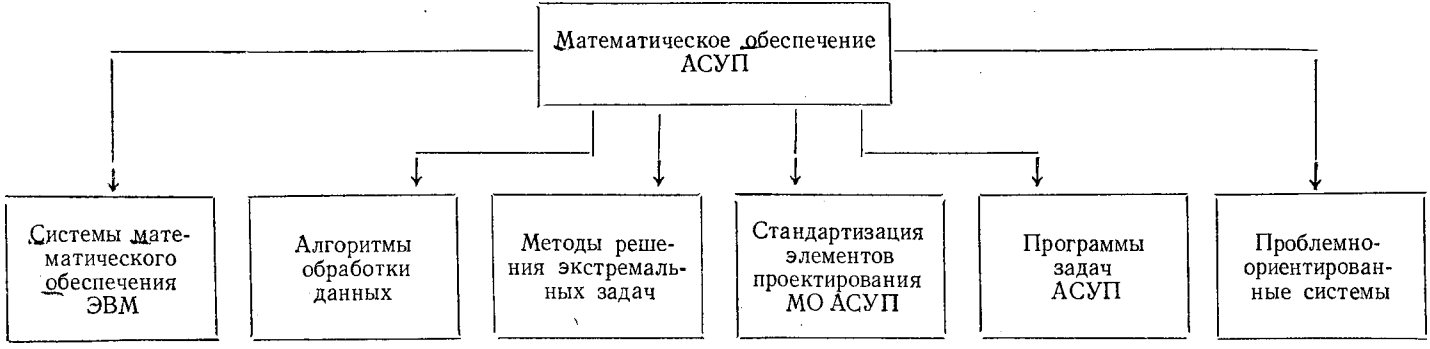


Рис. 1.1. Общая структура математического обеспечения АСУП

ной квалификации. Стандартизация инструкций по отладке, перфорации и решению задачи, стандартизация обозначений на блок-схемах, процессов обработки данных и языков программирования являются существенными вопросами в системе разработки АСУП. Отметим, что внедренная АСУП в первую очередь представляет сложный технологический процесс по сбору, обработке, передаче и хранению массивов информации. Необходима стандартизация элементов этого процесса, особенно в условиях широкого развертывания работ по разработке и внедрению АСУП.

Пятый класс — программы задач АСУП. На основе систем математического обеспечения ЭВМ, алгоритмов обработки данных, методов решения экстремальных задач и системы стандартизации элементов проектирования математического обеспечения АСУП разрабатываются программы для решения задач АСУП. Большое значение при построении этих программ имеет информационная увязка задач, согласование структур используемых и получаемых информационных массивов.

Шестой класс — проблемно-ориентированные системы. В общей проблеме разработки АСУП важным направлением является автоматизация процесса проектирования, заключающаяся в разработке комплекса постановок задач, алгоритмов и программ, имеющих возможность привязки к конкретным особенностям предприятий. Конечным результатом этого направления может быть построение некоторой системы автоматизации проектирования АСУП, включающей некоторый проблемно-ориентированный язык. Тогда работу по проектированию АСУП можно представить в следующем виде. При проведении обследования предприятия необходимо найти значения ряда параметров, характеризующих его информационно-экономическую систему. Значения этих параметров будут введены в память ЭВМ и обработаны по специальным программам проектирования математического обеспечения АСУП. В результате на основании использования специальной библиотеки алгоритмов и программ должны быть получены программы и инструктивные материалы, необходимые для внедрения и функционирования АСУП на этом предприятии.

Математическое обеспечение конкретной АСУП создается на основе использования внутреннего математического обеспечения ЭВМ, при этом используются трансляторы с алгоритмических языков, библиотеки стандартных программ, система организации информационных массивов, операционная система и др. Результатом этой работы являются алгоритмы и программы конкретных задач, составляющие основу основ АСУ.

Анализ распределения затрат на разработку целого ряда автоматизированных систем управления, которая проводилась в Центральном научно-исследовательском и проектно-технологическом институте организации и техники управления (ЦНИИТУ), показывает, что затраты на математическое обеспечение в зависимости от этапа проектирования автоматизированной системы

управления составляют от 15 до 70% годовых затрат. В среднем по ЦНИИТУ затраты на разработку математического обеспечения составляют около 35% от общих затрат на разработку автоматизированных систем управления. Однако заметна тенденция к увеличению доли затрат на математическое обеспечение АСУП. Исходя из накопленного опыта как в нашей стране, так и за рубежом, эти затраты должны составлять около 50% от общих затрат на разработку и внедрение автоматизированных систем управления.

Рассмотрим более детально характеристику перечисленных выше классов математического обеспечения АСУП. Отметим, что уровень математического обеспечения существенно зависит от уровня используемых средств вычислительной техники.

В настоящее время можно выделить три поколения электронных вычислительных машин. Каждое из этих поколений характеризуется своим управлением используемых технических средств, элементной базой этих средств и соответственно надежностью систем переработки информации.

Первое поколение ЭВМ — это ламповые машины. По сравнению с ручными методами использование ЭВМ позволило увеличить скорость и эффективность переработки информации. Хранение программ внутри памяти ЭВМ привело к значительному увеличению скорости счета, сделало возможным многократное повторение вычислений с разными входными данными. Однако в силу примитивности средств программирования, недостаточной надежности устройств машин, высокой стоимости сфера их применения была весьма ограниченной.

Для ЭВМ первого поколения системы переработки информации характеризуются специализацией по классам задач и универсальностью в своем классе. Так, можно выделить ЭВМ для решения научно-технических задач, экономических задач, задач управления в реальном масштабе времени и др. Возможности системы по переработке информации на ЭВМ первого поколения ограничены в силу слабых технических характеристик оборудования машин и сложности программирования. Ручное управление прохождением программ через вычислительную машину еще более уменьшало эффективность использования оборудования.

Стремление расширить возможности систем переработки информации и увеличить их эффективность определило требования к ЭВМ второго поколения. Развитие элементной базы — появление полупроводниковых и магнитных элементов — позволило обеспечить новый уровень надежности работы оборудования.

Требование эффективной переработки информации привело к увеличению быстродействия и надежности работы оборудования, устранению многих моментов, ограничивающих использование этого быстродействия. Несоответствие скоростей работы механических и электронных узлов было устранено организацией параллельной работы. Существенно был автоматизирован процесс

прохождения задачи через ЭВМ. Широкое применение алгоритмических языков и соответствующих транслирующих систем позволило существенно уменьшить время подготовки программ и получения результатов расчета.

Система переработки информации на ЭВМ второго поколения характеризуется более сложной структурой. В ней появляется новый компонент — операционная система. Операционная система — это организационная совокупность приемов и процедур для работы на машине. На операционную систему были возложены функции по управлению прохождением задач через вычислительную систему, по организации ввода-вывода, по трансляции с алгоритмических языков.

Существенной предпосылкой к развитию вычислительных систем третьего поколения явилось появление новой элементной базы — интегральных модулей и микроферритов.

Стремление к удовлетворению разнообразных требований пользователей привело к созданию ряда программно-совместимых моделей, построенных по агрегатному принципу. Вырос ассортимент внешних устройств, обеспечивающих разнообразие применений вычислительной системы. Система переработки информации на ЭВМ третьего поколения обладает качественно новыми возможностями, выражающимися, например, в развитии систем разделения времени, обеспечивающих коллективную работу многих пользователей одновременно.

Возможности и эффективность переработки информации на вычислительной системе третьего поколения существенно выросли в силу наличия мощной операционной системы, которая вобрала в себя черты операционной системы ЭВМ второго поколения и обеспечила их дальнейшее развитие.

Существенной чертой вычислительных систем третьего поколения является то, что упор при обеспечении эффективности переработки информации делается на развитие операционной системы. Это в свою очередь обеспечивает автоматизацию функций пользователя, подготавливающего задачу к решению на ЭВМ и организующего прохождение ее через машину.

Использование параллелизма работы устройств ЭВМ третьего поколения привело к асинхронной структуре машины. Структура ЭВМ является иерархической в силу принципа подчиненности оборудования, необходимого для согласования различных скоростей устройств и переработки потоков информации различной плотности. Информационные потоки в асинхронной системе замыкаются через оперативную память.

1.2. ХАРАКТЕРИСТИКА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ЭВМ

Приведем некоторые данные, характеризующие затраты на разработку систем математического обеспечения ЭВМ (СМО ЭВМ). В табл. 1.1 приведено соотношение затрат на разработку

внутреннего математического обеспечения и разработку техники по ряду фирм США, занимающихся разработкой ЭВМ [32].

В 1965 г. в США из общих затрат на электронную вычислительную технику, составляющих 6 млрд. долларов, на математическое обеспечение было израсходовано 3,2 млрд. долларов. Это составило немногим более 53% от общих затрат. В 1970 г. общие расходы на электронную вычислительную технику в США составили около 12 млрд. долларов, из них около 7 млрд. израсходованы на разработку математического обеспечения этой техники.

Из приведенных данных явно видна тенденция к возрастанию затрат на математическое обеспечение ЭВМ.

Таблица 1.1

Год	% затрат	
	Математическое обеспечение (Software)	Техника (Hardware)
1952	15	85
1964	42	58
1966	51	49
1970	58	42

По данным американской фирмы ИБМ [31], системная библиотека модулей, составляющих внутреннее математическое обеспечение ЭВМ типа ИБМ/360 и позволяющих создание операционной системы ОС/360, занимает около 10 млн. байтов, или более 2 млн. команд. На ее разработку потребовалось около тысячи человеко-лет. Эта система разрабатывалась в течение трех лет, при этом ежегодно тратилось около 130 млн. долларов.

Анализ структуры затрат на производство вычислительных работ в США показывает, что и в этом случае наблюдается падение удельного веса затрат на оборудование и повышение доли затрат на разработку и сопровождение математического обеспечения. Ниже приведена сложившаяся к 1969 г. структура затрат [14]:

— стоимость оборудования 35%
— сопровождение программного хозяйства 15%
— эксплуатация оборудования, операторская служба, перфорация 30%
— разработка новых программ 20%

Фирма ИБМ ввела [14] разделение комплексных услуг, предоставляющихся пользователю, на отдельные виды обслуживания с раздельными ценами на каждый вид и с предоставлением пользователю возможности применения только части этих услуг. К таким видам услуг относятся следующие:

1) продажа или аренда оборудования только вместе с внутренним программным обеспечением (супервизор, канальные программы и т. п.);

2) эксплуатация оборудования;

3) продажа права на пользование так называемыми «программными изделиями», к которым относятся все прикладные комплекты программ, все виды средств программирования (трансляторы, генераторы и т. п.), а также такие виды служебных программ, как программы работы с информационными массивами. При этом гарантируется сопровождение программных изделий;

4) услуги системных инженеров по проектированию или выбору специфической системы обработки данных, необходимой пользователю;

5) подготовка и обучение персонала пользователя.

В США существует целый ряд фирм (количество этих фирм все время возрастает), которые специализируются только на разработке математического обеспечения ЭВМ. При этом эти фирмы занимаются разработкой как внутреннего математического обеспечения, так и прикладных программ, в том числе и систем обработки данных для конкретных фирм.

Аналогичная тенденция к росту затрат на внутреннее математическое обеспечение ЭВМ наблюдается в нашей стране.

Отсюда можно сделать вывод, что разработка математического обеспечения становится развитой и самостоятельной по существу отраслью, безусловно являющейся решающей силой дальнейшего развития применения средств вычислительной техники.

Под системой математического обеспечения ЭВМ понимается некоторая операционная система (рис. 1.2), состоящая из двух основных частей: управляющей и обрабатывающей систем.

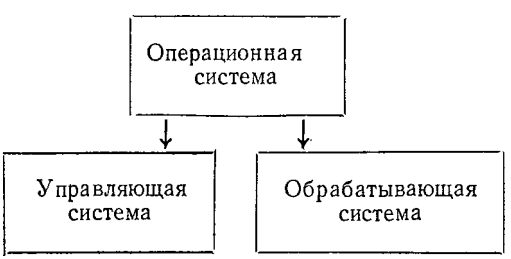


Рис. 1.2. Состав операционной системы

Операционная система служит для организации эффективного использования оборудования вычислительной системы, она предоставляет широкий ассортимент услуг для обеспечения разнообразных применений ЭВМ и способов работы. Наличие операционной системы позволяет автоматизировать многие неэффективные действия человека-оператора, в результате чего эффективность системы переработки информации повышается. На

операционную систему возлагаются функции транслирования, редактирования, компоновки и отладки программ. Операционная система организует управление и контроль за прохождением задачи, помогает оператору обслуживать устройство вычислительной машины.

Можно выделить три основные функции по управлению вычислительным процессом:

1. Управление данными, включающими организацию доступа к массивам информации и изменению массивов информации, хранящихся на различных внешних устройствах ЭВМ.

2. Управление заданиями, заключающееся в формировании потока задач в вычислительной системе.

3. Управление задачами, заключающееся в организации процесса выполнения потока задач в вычислительной системе и обмене информацией с оператором.

Структура составных частей операционной системы является существенным фактором, определяющим, насколько значительной должна быть переработка операционной системы в случае необходимости ее изменения, что в конечном итоге определяет жизнеспособность такой системы.

Операционная система, оформленная в виде единой программы, трудно расшифровывается и не обладает возможностью роста и изменения. В принципе это замечание верно для любой достаточно сложной и большой программы. Поэтому возникает необходимость дифференциации и специализации функций операционной системы и оформления их в виде отдельных функционально-автономных блоков. Однако выделение таких блоков не решает проблему роста и изменения операционной системы, поскольку стыковка этих блоков может оставаться достаточно сложной. Отсюда возникает второе требование к структуре компонентов операционной системы — стандартизация связей между блоками.

В результате можно сформулировать основной принцип построения операционной системы, заключающийся в выделении отдельных функций и оформлении их в виде отдельных стандартизованных блоков, функционирование которых зависит от значения входов и выходов и не зависит от функционирования других блоков. Такой принцип называется модульным, а программный блок, реализующий определенную функциональную возможность и рассчитанный на стандартные формы связи,— программным модулем.

Рассмотрим характеристику составных частей операционной системы. Структура управляющей системы приведена на рис. 1.3.

Диспетчер заданий предназначен для ввода входного потока заданий, выделения шагов заданий, назначения устройств ввода-вывода через диспетчер ввода-вывода и выделения оперативной памяти для шага задания. Существует несколько видов диспетчеров, которые задают различные режимы работы системы. Мож-

Рис. 1.3. Структура управляющей системы

но выделить два основных режима диспетчеризации — пакетной обработки и приоритетной обработки. В первом режиме задания выполняются по мере поступления в систему, во втором — по принципу назначенных или полученных приоритетов. Задания записываются на специальном языке управления заданиями, включающем несколько управляющих операторов.

Супервизор координирует поток задач через систему, распределяет между задачами ресурсы системы (время центрального процессора, оперативную память, устройства ввода-вывода и т. п.), планирует все операции ввода-вывода, получая управление через прерывание. Супервизор делится на супервизор задач и супервизор ввода-вывода.

Супервизор задач задает режим мультипрограммного выполнения задач, синхронизирует выполнение программ с операциями ввода-вывода и завершением других операций, регистрирует ошибки в программах, реализует некоторые процедуры выхода из аварийных ситуаций, осуществляет обслуживание по таймеру (системным часам).

Супервизор ввода-вывода запоминает и выполняет требования по вводу-выводу информации.

Главный диспетчер обрабатывает команды оператора и выдает сообщения вычислительной системы на пульт оператора. Для этих целей существует специальный язык диалога вычислительной системы и оператора.

Система управления данными включает язык макрокоманд ввода-вывода и соответствующие программы по управлению данными.

Обслуживающие программы позволяют организовать различные этапы подготовки программ к выполнению на машине. Программу задачи программист может разбить на логически независимые части — исходные модули, которые могут быть записаны на различных алгоритмических языках. Исходные модули после ввода в машину помещаются в библиотеку исходных модулей. После трансляции получается рабочий модуль, представляющий программу, записанную на машинном языке, но еще не готовую к выполнению.

С помощью редактора связи (редактора-компоновщика) программа может быть превращена в загрузочный модуль, готовый к выполнению на машине. Из таких модулей компонуется библиотека загрузочных модулей. Редактор связи может компоновать загрузочный модуль из рабочих модулей или использовать ранее подготовленный модуль.

Под отладчиками понимается система программ, реализующая моделирование вычислительного процесса с целью проверки соответствия отлаживаемой программы заложенному в ней алгоритму расчета.

Так как программы на языке загрузки, как правило, имеют свой язык, отличный от внутреннего языка ЭВМ, то для приве-

дения таких программ к виду, пригодному для исполнения, используются загрузчики. Для организации и ведения библиотек программ на разных языках используется библиотекарь. Эти программы и составляют систему преобразования программ.

Обрабатывающую систему по составу можно разделить на следующие три основные части (рис. 1.4): библиотеку стандартных программ, библиотеку генераторов и набор трансляторов.

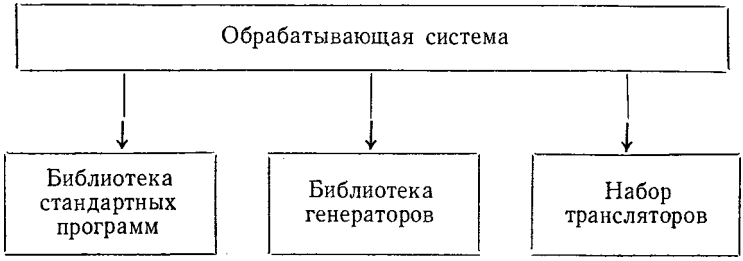


Рис. 1.4. Состав обрабатывающей системы

Библиотека стандартных программ (рис. 1.5) может быть ориентирована на решение некоторого класса задач. В библиотеку включаются оформленные по определенным правилам программы, готовые для использования при составлении программ конкретных задач. В библиотеку стандартных программ могут входить программы, реализующие вычисление элементарных функций, переводы из одной системы счисления в другую, решение систем дифференциальных и интегральных уравнений, операции

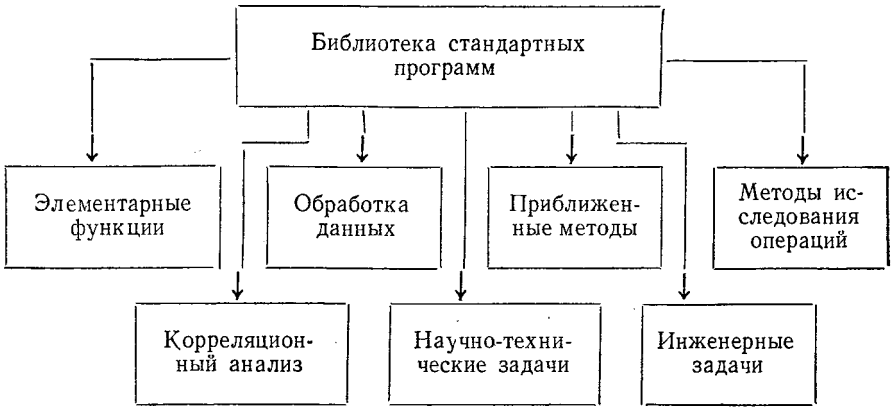


Рис. 1.5. Состав библиотеки стандартных программ

над матрицами и векторами, обработку больших массивов информаций, включающую предварительную обработку информации, сортировку информации, корректировку информации, под-

готовку информации к выводу на внешние носители и другие типовые процессы обработки, решение задач линейного и динамического программирования, решение задач по технологическому и конструкторскому проектированию и др.

Под генератором программ понимается программа, которая на основании значений параметров, характеризующих массивы входной и выходной информации, некоторых указаний пользователя и свободных ресурсов вычислительной системы (оперативная память, магнитные ленты, устройства ввода-вывода и т. п.) создает программу, реализующую заданную обработку информации и наиболее эффективно использующую имеющиеся ресурсы вычислительной системы.

На рис. 1.6 приведена принципиальная блок-схема работы генератора программ обработки данных.

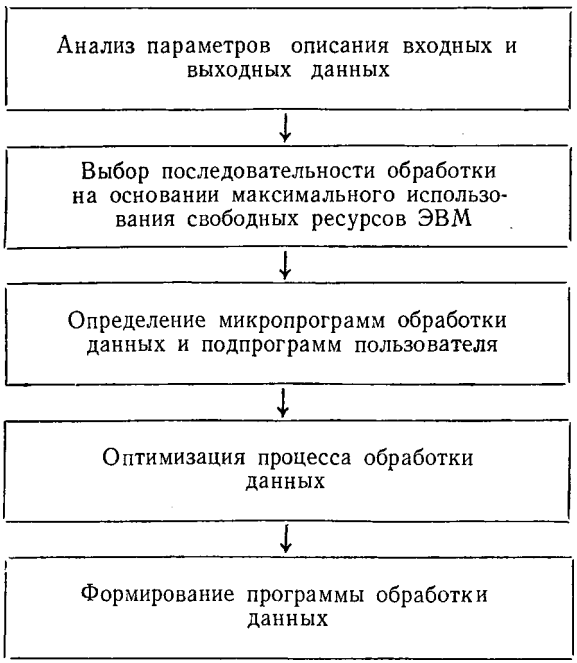


Рис. 1.6. Принципиальная блок-схема работы генератора программ обработки данных

Приведем краткую характеристику некоторых генераторов, которые можно эффективно использовать при разработке программ для решения задач из класса АСУП. Генератор ввода предназначен для ввода информации с разных носителей, логического контроля этой информации, компоновки и записи обработанной информации на внешние носители. Генератор сортировок

предназначен для сортировки информации. Существенно важным блоком этого генератора является блок выбора наиболее подходящего алгоритма сортировки в зависимости от длины и структуры входного массива. Генератор поиска предназначен для поиска информации в разных массивах. Генератор массивов осуществляет получение выходных массивов из входных при условии, если входные массивы соответствующим образом упорядочены. Генератор отчетов осуществляет обработку массивов, подготовку, редактирование и вывод разных форм результатной информации. Генератор разузлования на основании информации о структуре изделия, представленной в виде графа, получает массивы, в которых приведена рассчитанная общая применяемость деталей (узлов) в заданном наборе изделий. Генератор задач на основании описания форм входных и результатных документов, описания структуры входных, промежуточных и выходных массивов и схемы вычислительного процесса, описанного на уровне типовых процессов обработки данных, составляет программу решения соответствующей задачи и формирует пакет программ решения этой задачи. В принципе набор генераторов может быть существенно расширен.

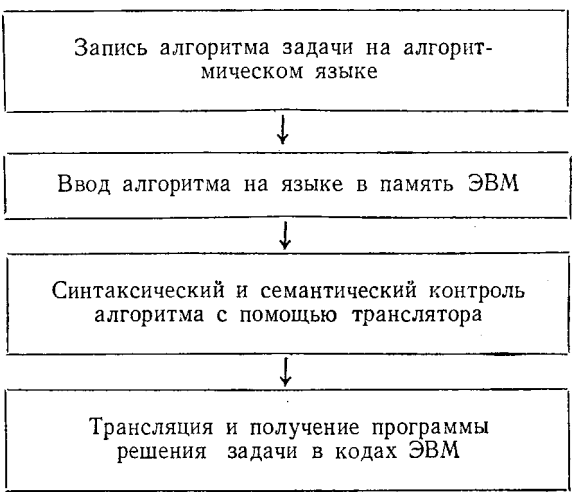


Рис. 1.7. Схема применения алгоритмических языков для составления программ задач

В общем случае под транслятором понимается программа, которая алгоритм некоторой задачи, записанный на алгоритмическом языке, переводит на другой язык, например, на язык команд ЭВМ.

Под алгоритмическим языком понимается некоторый формализованный язык, предназначенный для записи алгоритмов об-

Рис. 1.8. Классификация алгоритмических языков

работки информации и ориентированный на решение определенного класса задач. Основное назначение алгоритмических языков — это упрощение процесса составления программ для ЭВМ. Применение алгоритмических языков является одним из способов решения проблемы программной совместимости различных ЭВМ. Алгоритмический язык, с одной стороны, настолько приближен к языку специалиста, что позволяет понимать алгоритм задачи, написанной на этом языке; с другой стороны, он настолько формализован, что соответствующая программа может быть воспринята ЭВМ. Для такой расшифровки алгоритма, записанного на алгоритмическом языке, и перевода его в коды конкретной ЭВМ предназначены трансляторы. На рис. 1.7 приведена схема получения рабочей программы в кодах ЭВМ при использовании алгоритмических языков.

Записанный на языке алгоритм решения задачи (программа на алгоритмическом языке) переносится на машинные носители (перфоленту или перфокарты) и вводится в память ЭВМ. На первом этапе работы транслятора осуществляется синтаксический и семантический контроль программы на алгоритмическом языке. При обнаружении ошибок выдается соответствующая информация, анализируя которую, программист должен исправить обнаруженные ошибки. Примерами ошибок могут быть такие: закрывающих скобок в некоторой формуле меньше, чем открывающих; в вычислениях используется переменная, которой не присвоено значение; пропущены некоторые знаки пунктуации и т. п. Исправленная программа транслируется и получается рабочая программа в кодах ЭВМ.

Алгоритмические языки можно разделить на три больших класса (рис. 1.8): машинно-ориентированные, процедурно-ориентированные и проблемно-ориентированные.

Машинно-ориентированные языки относятся к языкам типа 1:1, т. е. языкам, для которых, как правило, из одной команды на входном языке получается одна команда на машинном языке. Такими языками являются язык символического кодирования (ЯСК) для ЭВМ «Минск-22» и «Минск-32» [23], ассемблер для ИБМ/360 [9] и др.

Некоторым расширением машинно-ориентированных языков являются автокоды и макроассемблеры. Под макроассемблером понимается некоторый ассемблер с изменяемым набором макрокоманд.

Примером автокода может служить универсальный машинно-ориентированный язык АЛМО [17]. Язык АЛМО — это язык некоторой гипотетической машины, достаточно близкий языкам конкретных ЭВМ. Этот язык может служить в качестве языка-посредника. На рис. 1.9 приведена схема использования промежуточного языка АЛМО. Программа составляется на некотором языке более высокого класса или непосредственно на АЛМО, затем транслируется на промежуточный язык АЛМО (этот этап

может не выполняться, если программа уже составлена на АЛМО) и только после этого — в коды конкретной ЭВМ.

В случае большого многообразия языков машинных команд количество трансляторов может быть сильно сокращено за счет использования языка-посредника АЛМО. При наличии m машинных и n алгоритмических языков нужно $m \cdot n$ прямых трансляторов для перевода программы с алгоритмического языка на язык ЭВМ и $m+n$ трансляторов в случае использования языка-посредника, но при этом исходная программа должна транслироваться дважды.

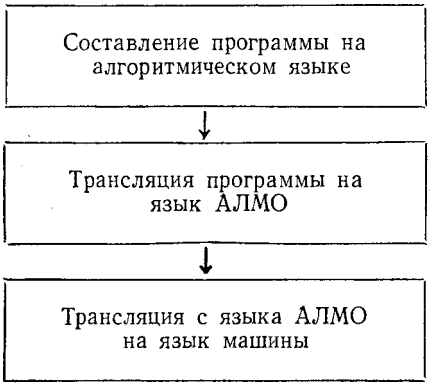


Рис. 1.9. Схема применения машинно-ориентированного языка АЛМО

Процедурно-ориентированные языки — это класс алгоритмических языков, операторы которых имеют некоторую проблемную ориентацию.

Среди множества процедурно-ориентированных экономических языков можно назвать такие, как КОБОЛ [10, 9], АЛГЭК [20]. Эти языки имеют достаточно мощные средства для записи алгоритмов по обработке больших массивов информации и как следствие приспособлены для решения экономических задач, задач из класса АСУП. Для решения научно-технических задач широкое распространение получили алгоритмические языки АЛГОЛ [25], ФОРТРАН [37], АЛГАМС [28] и др.

К классу универсальных алгоритмических языков, имеющих достаточно развитый процедурный аппарат для решения задач разных классов, в том числе экономических, научно-технических, информационных (построения информационно-поисковых систем), можно отнести ПЛ/1 [36] и АЛГОЛ-68 [2]. Преимущество таких языков, являющееся следствием применения единого алгоритмического языка,— возможность использования программ, составленных и опробованных на разных машинах, и упрощение процесса обучения программистов (отпадает необходимость специализации по разным классам задач). Но наряду с этим приме-

нение универсальных алгоритмических языков имеет ряд недостатков. Во-первых, эти языки достаточно сложны; в результате вопрос изучения и применения этих языков специалистами становится весьма трудным. Во-вторых, трансляторы с этих языков представляют достаточно сложную систему программ и, как следствие, их разработка требует больших затрат. Кроме того, эти трансляторы могут работать достаточно эффективно только на больших ЭВМ. В-третьих, из-за сложности трансляторов остается проблематичным вопрос об эффективности полученных после трансляции рабочих программ.

В настоящее время насчитывается несколько тысяч алгоритмических языков, ориентированных на решение задач разных классов и на разные ЭВМ. Из этого многообразия большую часть составляют специализированные языки, ориентированные на решение задач некоторого узкого класса.

Так, для составления программ для задач из таких областей, как создание информационно-поисковых систем, моделирование процессов познания, автоматический перевод, некоторые вопросы исследования операций и создание искусственного интеллекта, используются языки ЛИСП, ИПЛ, СЛИП, КОМИТ [33] и др.

Некоторое представление о применении различных алгоритмических языков дают следующие данные об использовании в 1968 г. средств программирования на машинах, принадлежащих государственным учреждениям США [14]: ассемблеры — 48%, КОБОЛ — 29%, ФОРТРАН — 12%, генератор отчетов — 9%, ПЛ/1 — 1%, машинные языки — 1%.

Следует отметить, что среди процедурно-ориентированных языков КОБОЛ занимает ведущее положение (69% в этом классе). Такое положение сохраняется и в настоящее время.

Проблемно-ориентированные языки предназначены для записи системы задач из определенной области, например, для проектирования АСУ, проектирования технологий, конструкторского проектирования и др. С одной стороны, проблемно-ориентированные языки синтаксически должны быть проще, чем процедурно-ориентированные. С другой стороны, проблемно-ориентированный язык может опираться на некоторый процедурно-ориентированный язык.

Использование проблемно-ориентированного языка можно представить в виде следующей последовательности: составление программ комплекса задач на этом языке, трансляция на промежуточный процедурно-ориентированный язык с последующей трансляцией на машинный язык.

1.3. ХАРАКТЕРИСТИКА АЛГОРИТМОВ ОБРАБОТКИ ДАННЫХ

Возможности вычислительной системы и характер обработки информации в конкретной задаче определяют организационную структуру используемых информационных массивов.

Можно выделить следующие способы организации структур информационных массивов: линейные, многоуровневые, списковые.

При использовании магнитной ленты в качестве основной внешней памяти ЭВМ наиболее распространен линейный метод организации информационных массивов. При этом способе из общей совокупности информации выделяются отдельные массивы. Каждый из массивов может иметь свою внутреннюю структуру и некоторые особенности организации. В то же время все массивы должны подчиняться общим законам организации подобного рода массивов. В свою очередь массивы состоят из более мелких информационных образований, из так называемых записей или машинных документов (на рис. 1.10 приведена схема такой подчиненности).

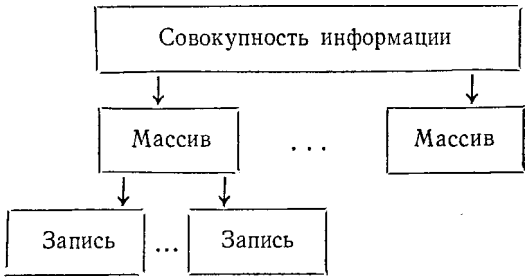


Рис. 1.10. Схема подчиненности информационных совокупностей

Записи состоят из реквизитов. Значение каждого реквизита может быть представлено в каком-то коде и системе счисления. По структуре записи массивы можно разделить на два больших класса: с постоянной длиной и с переменной длиной записей. Массив первого класса состоит из записей одной и той же длины. Массив второго класса может содержать записи разной длины. При этом структура записей переменной длины может быть достаточно сложной.

Одним из примеров многоуровневой организации информационного массива может быть применение каталога. Информационный массив в этом случае может состоять как из записей постоянной длины, так и переменной, а структура записи может представлять многоуровневую совокупность. Массив разбивается на сегменты, причем длина сегмента должна быть такой, чтобы он весь мог поместиться в отведенное место оперативной памяти. Обязательным дополнительным условием является то, что этот массив должен быть рассортирован по определенным реквизитам-признакам. Реквизиты-признаки первой записи в каждом сегменте вместе с адресом этой записи на магнитной ленте составят новый массив-каталог. Поиск необходимой информации

(рис. 1.11) при этом осуществляется многоступенчато. По значению исходных реквизитов-признаков в массиве-каталоге, который должен быть размещен в оперативной памяти, отыскивается начало соответствующего сегмента на магнитной ленте. Затем в оперативную память вызывается информация данного сегмента и только после этого осуществляется поиск информации внутри сегмента, расположенного в оперативной памяти.

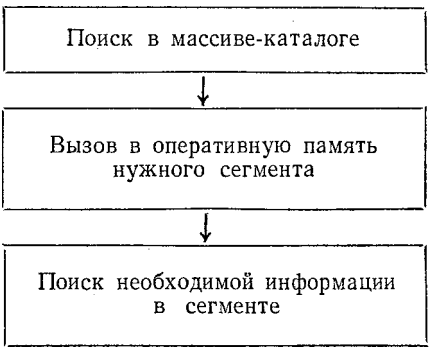


Рис. 1.11. Схема поиска в массиве с каталогом

Отметим, что массив-каталог может быть достаточно длинным и для поиска в нем можно организовать каталог более высокого порядка.

Списковые структуры — это такая организация массива, при которой связь между отдельными информационными совокупностями устанавливается с помощью специальных ссылок.

Собственно алгоритмы обработки данных можно разбить на два класса: алгоритмы организации информационных массивов и алгоритмы использования этих массивов.

Анализ алгоритмов обработки экономической информации позволил выделить следующие типовые процессы обработки:

— обмен информацией между оперативной памятью и внешней;
— обмен информацией между разными уровнями оперативной памяти;
— редактирование информации;
— преобразование массивов информации;
— преобразование внутренней структуры массивов информации;
— ввод информации с промежуточных носителей (перфоленты, перфокарты) с контролем;
— приведение информации к стандартной структуре (компоновка);
— сортировка информации;
— корректировка информации;
— вывод информации на внешние носители;

— поиск необходимых показателей в массивах информации;
— получение итоговых данных в виде самостоятельных массивов;
— управление массивами, обеспечивающее возможность доступа к данным, вызов для обработки очередной информационной совокупности из массива;
— изменение форматов данных;
— выборка информации;
— обработка информации, представленной в виде матричных таблиц.

Алгоритмы, реализующие выполнение соответствующих типовых процессов обработки информации, существенно зависят от принятой организационной структуры информационных массивов.

1.4. ВОПРОСЫ СТАНДАРТИЗАЦИИ МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ

Система стандартизации* элементов проектирования математического обеспечения АСУП (рис. 1.12) включает единую символику описания алгоритмов и программ, терминологический словарь, регламентирующий терминологию специалистов, занимающихся разработкой и внедрением АСУП, стандартизацию этапов проектирования математического обеспечения АСУП и состава документации на каждом из этапов. Особое положение занимает технология обработки данных на ЭВМ. Здесь необходимы стандарты на маркировку носителей, систему подготовки, хранения и обновления носителей информации (перфокарты, перфоленты, магнитные ленты, дуаль-карты, бланки, магнитные диски и др.). Необходима система в инструкциях персоналу, регламентирующих процессы перенесения программ и данных на машинные носители, процессы отладки и решения задач, накопление и обработку статистических данных о подготовке и решении задач.

1.5. ХАРАКТЕРИСТИКА ЗАДАЧ АСУП

По функциональному назначению автоматизированную систему управления предприятием можно разбить на ряд подсистем, основными из которых являются:
— организация и ведение массивов нормативно-справочной информации;
— техническая подготовка производства;
— технико-экономическое планирование;
— оперативно-календарное планирование;
— управление материально-техническим снабжением;
— бухгалтерский учет;
— управление реализацией и сбытом продукции.

---

* Этим вопросам посвящена книга Г. Х. Брандона [7].

Рис. 1.12. Классификация стандартов на математическое обеспечение АСУП

Естественно, что все эти подсистемы информационно взаимосвязаны и представляют некоторую модель информационно-экономической системы предприятия.

С точки зрения информационных взаимосвязей и обеспечения информацией всех подсистем наиболее важной является подсистема организации и ведения массивов нормативно-справочной информации. Основными массивами нормативно-справочной информации на предприятии с дискретным характером производства являются конструкторский массив, раскрывающий состав изделий, массив материальных нормативов, определяющий потребность материалов на изготовление деталей (узлов), массив трудовых нормативов, определяющий пооперационную трудоемкость деталей (узлов), классификатор-ценник материалов и комплектующих изделий, массив основных средств, содержащий информацию о наличии оборудования на предприятии, массив личного состава, содержащий информацию о работниках предприятия, и др.

В качестве основного машинного носителя нормативно-справочной информации используется магнитная лента.

Функционально организацию и ведение массивов нормативно-справочной информации можно подразделить на два этапа: организацию массивов на магнитных лентах и ведение массивов на магнитных лентах.

Под организацией массивов на магнитных лентах понимается разовая работа по перенесению информации с машинных носителей (перфолент или перфокарт) на магнитные ленты, приведение массивов к виду, удобному для последующего использования, проведение всевозможных работ по выверке комплектности и достоверности информации.

На рис. 1.13 приведена принципиальная схема начальной организации массива. В блоке *1* осуществляется ввод исходной информации с перфолент и перфокарт. Каждая введенная порция информации подвергается логическому контролю. Один из видов логического контроля — проверка значения каждого реквизита на соответствие минимальной и максимальной границе изменения соответствующего реквизита. Кроме того, при этом контроле проверяется схема перфорации соответствующих исходных документов. Такие ошибки, как пропуск реквизита или появление лишнего, с помощью такого контроля легко улавливаются.

Обнаруженный неправильный документ выбрасывается из дальнейшей обработки. Соответствующая печать сигнализирует о наличии ошибок в отперфорированной информации. Правильная информация подвергается дальнейшему преобразованию (компоновке) и представляется в виде совокупности записей. Во время компоновки могут осуществляться: перевод значений реквизитов в другую систему счисления, изменение их форматов, подстановка вместо некоторых шифров каких-то совокупностей других шифров или соответствующих этим шифрам наименова-

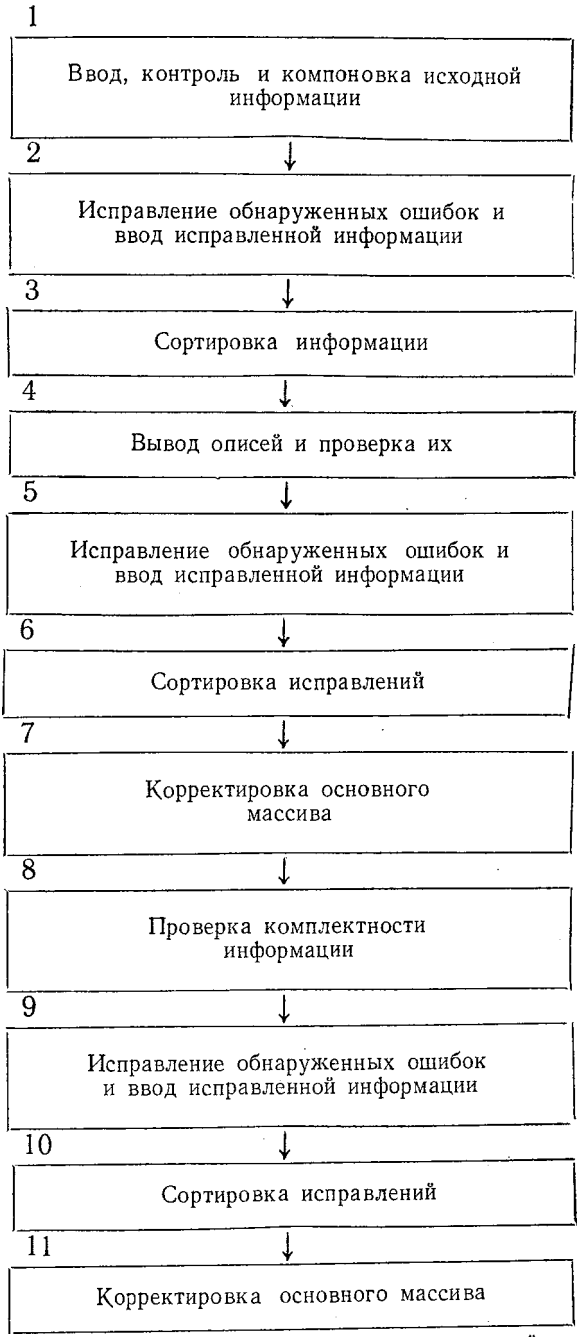


Рис. 1.13. Принципиальная схема начальной организации массива нормативно-справочной информации

ний каких-то объектов. В результате работы этого блока на магнитных лентах будет записана совокупность записей соответствующего массива.

В блоке *2* обнаруженные во время работы программы логического контроля неправильные данные после их исправления и повторной перфорации вводятся в память машины и присоединяются к полученному в блоке *1* массиву. Для ввода исправленной информации может использоваться та же программа ввода, логического контроля и компоновки информации, что и в блоке *1*.

В блоке *3* полученная ранее информация для удобства как дальнейшего использования в задачах, так и реализации процесса корректировки на этапах ведения массивов нормативно-справочной информации на магнитных лентах должна быть рассортирована по некоторой совокупности реквизитов-признаков. Полученный массив будем называть основным.

В блоке *4* осуществляется вывод описей соответствующего массива информации. Смысл работы этого блока заключается в том, что на бумагу переносится вся информация, записанная и рассортированная на магнитных лентах (эти документы будем называть описями). Описи являются упорядоченной копией исходной для данных массивов информации. Их назначение состоит в обеспечении возможности дополнительной выверки правильности полученной на магнитных лентах информации. Кроме того, описи удобно применять для выяснения возможных недоразумений при дальнейшем использовании массивов информации. Целесообразность получения описей массивов нормативно-справочной информации подтверждается многолетним опытом ведения постоянных карточек на машиносчетных станциях.

Обнаруженные после проверки описей ошибки должны быть исправлены, перенесены на машинные носители и введены в память машины. Эти функции выполняются блоком *5*. При этом используется та же программа ввода, логического контроля и компоновки, что и в блоке *1*.

В блоке *6* производится сортировка введенной в предыдущем блоке информации. Сортировка этой информации должна осуществляться по тем же реквизитам-признакам, что и информация основного массива.

В блоке *7* происходит корректировка основного массива. В условиях, когда в качестве основного носителя информации используется память с последовательным доступом (магнитная лента), целесообразно использовать принцип корректировки, известный под названием «отец-сын» [11]. Принцип этот заключается в следующем. На магнитных лентах должен быть записан основной массив информации («отец»), затем записываются корректуры. Корректуры должны быть упорядочены по тем же реквизитам-признакам, что и основной массив. Эти два массива просматриваются синхронно. В результате такого просмотра и последовательного сравнения реквизитов-признаков из записей

одного и другого массивов на новых магнитных лентах формируется обновленный основной массив («сын»).

Целесообразность применения этого принципа определяется следующими факторами: его программная реализация наиболее проста, а применение гарантирует сохранность информации старого поколения от случайных сбоев. В случае использования магнитной ленты время работы машины по корректировке информации не очень велико. Если информация нового основного массива по каким-то причинам оказалась испорченной, то при условии сохранения магнитных лент с информацией старого основного массива процесс корректировки может быть достаточно легко повторен и информация восстановлена.

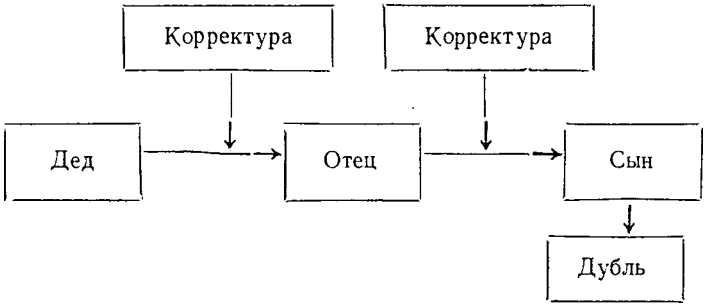


Рис. 1.14. Схема ведения массивов нормативно-справочной информации в трех поколениях

Отсюда следует, что, кроме обновленного основного массива, необходимо сохранить и информацию предыдущего поколения. Применяемая в ряде организаций схема «дед-отец-сын» является одним из возможных способов реализации принципа корректировки информации «отец-сын». Практика показывает, что сохранение информации трех последних поколений обеспечивает очень высокие гарантии по восстановлению последнего, основного массива. На рис. 1.14 приведена схема ведения массивов нормативно-справочной информации в трех поколениях. Из схемы следует, что необходимо сохранить третье поколение («дед») и корректуру к нему, второе поколение («отец») и корректуру к нему и последнее поколение («сын»). Последнее поколение является рабочим вариантом основного массива. Эта информация используется при проведении всевозможных расчетов при решении задач. Для наиболее быстрого восстановления основного массива целесообразно иметь дубль этого массива.

Как показывает анализ опыта организации массивов нормативно-справочной информации на промышленных предприятиях, одной из наиболее распространенных ошибок в этих массивах является некомплектность информации (например, пропущена ин-

формация по некоторым деталям или узлам). Выявление такого рода ошибок возложено на блок *8* (рис. 1.13). В зависимости от характера и назначения соответствующего массива эта проверка может осуществляться по-разному. Основным способом, применяемым для такого рода проверок, является проверка наличия соответствующих реквизитов-признаков в разных массивах. Например, в конструкторском массиве имеются шифры всех деталей (узлов), изготовляемых на данном предприятии. Эти же шифры должны быть в массиве трудовых нормативов. Такую проверку можно достаточно просто организовать с помощью соответствующих программ.

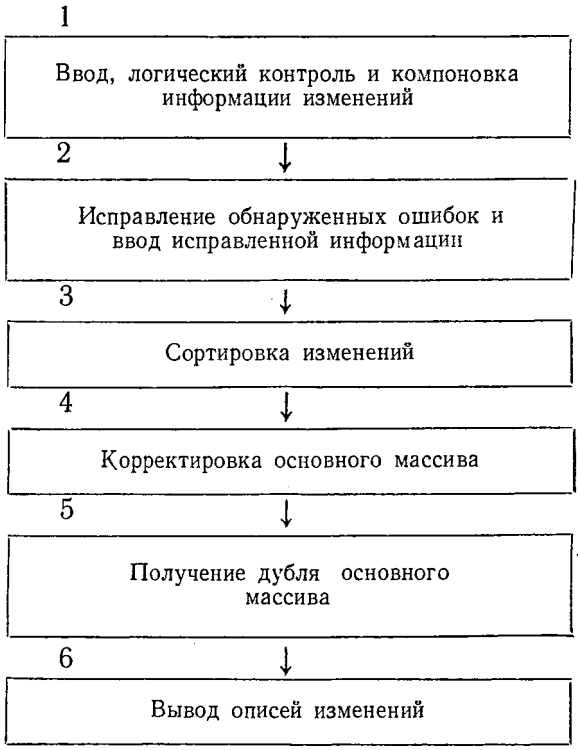


Рис. 1.15. Принципиальная схема корректировки массива нормативно-справочной информации

В результате работы программы блока *8* могут быть обнаружены ошибки в массиве. Обработка этих ошибок проводится аналогично обработке ошибок, обнаруженных после работы блока *4*. Другими словами, назначение и функции блоков *9—11* аналогичны назначению и функциям блоков *5—7*.

В результате проведения описанного комплекса работ на магнитных лентах будет получен основной массив, готовый для дальнейшего использования.

На рис. 1.15 приведена принципиальная схема реализации процессов ведения массива постоянной информации. Последовательность блоков, приведенных на этом рисунке, должна выполняться всякий раз, когда назревает необходимость проведения очередной корректировки массива.

В блоке *1* осуществляется ввод, логический контроль и компоновка информации изменений. Отметим, что формы документов и как следствие схемы перфорации для изменений могут отличаться от форм и схем перфорации основного массива. Одна из причин этого отличия — необходимость наличия в документах с изменениями номера извещения на изменение и информации о сроках ввода соответствующего извещения в действие. Кроме того, в изменениях может содержаться любая дополнительная информация.

Примем правило, по которому для аннулирования из основного массива какой-то записи достаточно в корректурах иметь запись с такими же основными реквизитами-признаками, как и в основном массиве, и с дополнительным реквизитом, определяющим вид корректировки. В качестве такого признака можно использовать равенство нулю одного из реквизитов-оснований. Используя это правило, можно легко осуществлять необходимые корректировки основного массива постоянной информации.

В блоке *2* обнаруженные во время работы программы ввода неправильные данные после повторной их перфорации вводятся в память машины и присоединяются к полученному в блоке *1* массиву. Работа этих блоков аналогична работе блоков *1* и *2* на принципиальной схеме начальной организации массива нормативно-справочной информации (рис. 1.13).

В блоке *3* производится сортировка изменений. При этом при наличии двух записей с совпадающими реквизитами-признаками эти записи должны располагаться в порядке увеличения срока ввода соответствующих извещений на изменение. В результате такого расположения при корректировке в основной массив будет отправлена запись с наиболее поздним сроком ввода в действие изменения.

В блоке *4* реализуется корректировка основного массива. В результате работы этого блока на магнитных лентах получается новый основной массив (согласно схеме на рис. 1.14 — «сын»). Магнитные ленты с информацией корректуры должны быть сохранены. Массив, ставший в результате этой корректировки предыдущим поколением («отец»), сохраняется только в одном экземпляре. Следовательно, один экземпляр этого поколения можно ликвидировать. Кроме того, необходимо сохранить массив, ставший после корректировки третьим поколением («дед»), вместе с массивом корректур. Массив, ставший после корректировки

четвертым поколением, вместе с массивом корректур можно ликвидировать. Таким образом происходит обновление массивов постоянной информации.

Как уже отмечалось, с нового основного массива нужно получить дубль. Это делается блоком *5*. При использовании основного массива может быть обнаружено, что он испорчен. Тогда необходимо организовать восстановление соответствующего поколения. Эта операция осуществляется или получением дубля, если это последнее поколение, или корректировкой предыдущего поколения.

В блоке *6* осуществляется вывод описей изменений для дополнительной проверки правильности перфорации и корректировки основной описи массива.

Полученные и поддерживаемые в рабочем состоянии массивы нормативно-справочной информации используются при решении задач других подсистем.

1.6. ПРОБЛЕМНО-ОРИЕНТИРОВАННЫЕ СИСТЕМЫ ПРОГРАММИРОВАНИЯ

Дальнейшим развитием направления по использованию алгоритмических языков является создание проблемно-ориентированных систем программирования. В приложении к вопросу проектирования АСУП построение проблемно-ориентированной системы заключается в создании системы описания объекта управления и некоторого алгоритмического языка, описывающего процесс обработки информации для соответствующего объекта. При этом объект управления может описываться в виде набора некоторых параметров. Примером набора таких параметров являются система планирования и учета на предприятии, форматы реквизитов, используемых в расчетах, периодичность решения отдельных задач, объемы входной, промежуточной и выходной информации, формы входных и результатных документов и др. На основании значений этих параметров и описанного процесса обработки информации должны быть выбраны необходимые элементы математического обеспечения, найдены связи между этими элементами, произведена информационная увязка, в результате должна быть построена некоторая система программ, реализующих обработку информации, необходимую для функционирования соответствующей автоматизированной системы управления.

В условиях широкого развертывания работ по созданию автоматизированных систем управления предприятиями актуальной задачей является автоматизация проектирования АСУП. Одним из направлений такой автоматизации является построение некоторой параметрической системы автоматизированного проектирования (ПСАП) АСУП.

Проект АСУП можно представить в виде некоторой логической функции σ, зависящей от множества параметров:

$$\sigma = f(p_1, p_2, \ldots, p_n).$$

Рис. 1.16. Структура математического обеспечения ПСАП АСУП

Определение перечня параметров и установление их функциональных зависимостей позволит формализовать процесс проектирования АСУП и как следствие создать комплекс программ, обеспечивающих автоматизацию проектирования АСУП (структура такого математического обеспечения приведена на рис. 1.16). Основой при разработке этих программ должна быть операционная система, включающая транслирующую систему с процедурно-ориентированного языка КОБОЛ, управляющую систему, библиотеку генераторов и стандартных программ, реализующих типовые процедуры обработки информации.

Библиотека модулей задач АСУП включает в себя систему организации и ведения информационной базы, модели задач, алгоритмы их решения и конкретные программы с указанием границ и условий их применения.

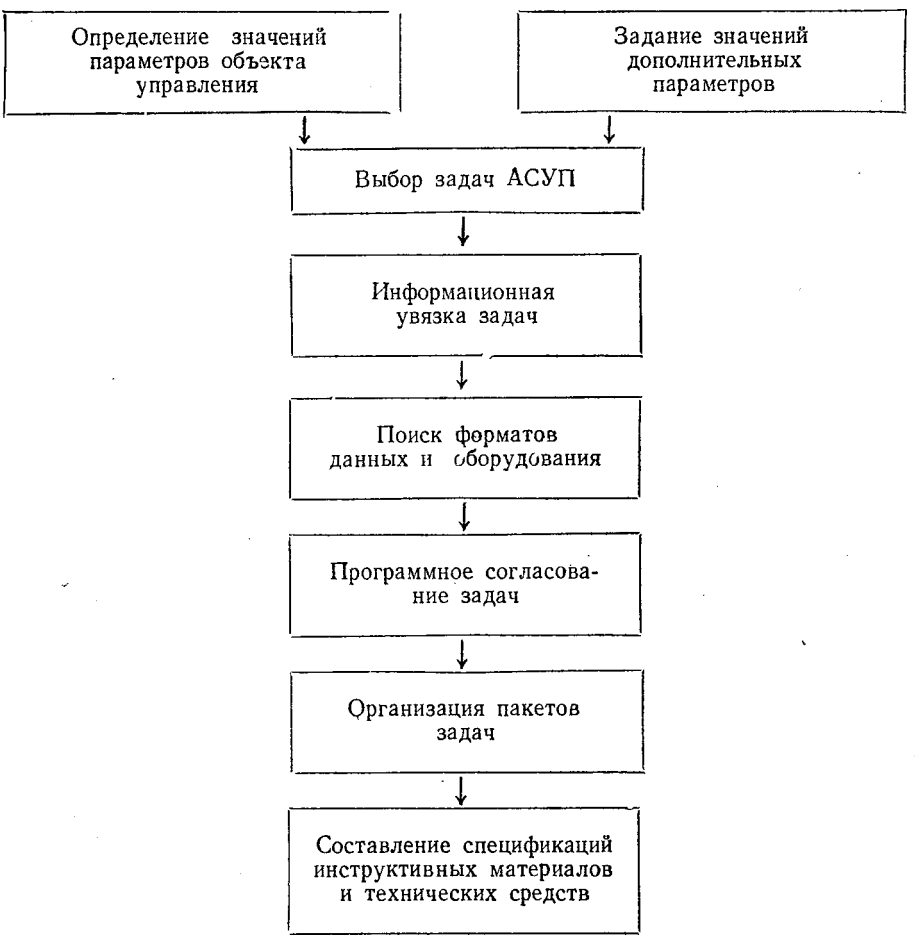


Рис. 1.17. Схема функционирования ПСАП АСУП

Система программ по организации, ведению и использованию вспомогательных массивов (рис. 1.16) предназначена для ведения (на основе библиотеки модулей задач АСУП и описания объекта управления) и организации массивов, содержащих информацию о форматах данных, структуре информационных массивов, параметрах объекта управления, об имеющемся в распоряжении пользователя оборудовании ЭВМ, о комплексе технических средств, об инструктивных материалах, необходимых при внедрении и функционировании АСУП, о спецификациях рабочей документации. Эти массивы предназначены для концентрации данных о задачах, массивах, технических средствах и инструктивных материалах. Такая организация массивов упрощает процессы привязки задач к конкретному объекту управления, согласования задач и построения системы обработки информации в АСУП.

Конечным результатом работы ПСАП должна быть взаимосвязанная и согласованная система программ работы вычислительной техники и инструктивные материалы для обслуживающего персонала при функционировании конкретной АСУП.

Согласование модулей и организация пакета задач обеспечиваются системой организации и управления пакетами задач АСУП.

Процесс функционирования ПСАП АСУП изображен на рис. 1.17. Он сводится к следующему. При обследовании необходимо определить значения каждого из параметров, характеризующих рассматриваемый объект управления. Используя значения полученных и введенных в память ЭВМ параметров, разработанная система алгоритмов выберет из библиотеки модулей необходимые разделы (программы или комплексы программ), произведет их настройку по этим параметрам, осуществит информационную увязку модулей и составит системную ленту с программами, ориентированными на решение определенных задач.

1.7. ТРЕБОВАНИЯ К СИСТЕМЕ ПРОГРАММИРОВАНИЯ ДЛЯ АСУП

Особенности задач из класса АСУП, типовые процессы обработки и анализ опыта обработки экономической информации на ЭВМ позволяют сформировать следующие требования к системе программирования, ориентированной на решение задач АСУП.

В системе математического обеспечения целесообразно иметь библиотеку стандартных программ, реализующих типовые процессы обработки данных. Причем для любого типового процесса должен быть или набор программ, эффективность каждой из которых зависит от параметров обрабатываемого массива, или универсальная программа, имеющая высокий коэффициент эффективности для достаточно большого разнообразия параметров обрабатываемых массивов, или генератор программ, который на основании значений параметров, характеризующих входные и выходные массивы, и свободных ресурсов вычислительной машины

строит достаточно эффективную программу, реализующую заданный процесс обработки.

В системе математического обеспечения необходима управляющая программа (диспетчер), реализующая управление ходом решения задач. Естественно, что возможности программы-диспетчера существенно зависят от параметров используемой ЭВМ. Одним из основных требований к этой программе является необходимость обмена информацией о ходе решения задач между ЭВМ и оператором, управляющим процессом решения. Для работы управляющей программы задачи могут быть оформлены в виде пакетов задач, для чего целесообразно иметь ряд программ, организующих оформление такого пакета и его корректировку.

Для автоматизированных систем управления предприятиями с дискретным характером производства управляющая программа должна обеспечивать прием и обработку дискретной информации в реальном масштабе времени.

В системе математического обеспечения должны быть сервисные программы, обеспечивающие отладку программ, внесение изменений в программы, организацию программных массивов в памяти ЭВМ и внешних носителях, оформление программной документации задач и т. п.

В системе математического обеспечения должны быть транслирующие системы, предназначенные для трансляции программ, записанных на алгоритмических языках, в коды конкретной ЭВМ. Набор таких систем существенно зависит от параметров используемой ЭВМ, класса решаемых задач, состава и качества библиотеки стандартных программ и ряда других факторов. Для задач из класса АСУП в качестве такого языка может быть использован алгоритмический язык КОБОЛ в совокупности с мнемокодом соответствующей машины.

Одним из недостатков применения развитых алгоритмических языков является то, что рабочие программы, полученные после трансляции, требуют больше времени для их выполнения и поэтому хуже, чем аналогичные программы, составленные программистом средней квалификации на машинно-ориентированном языке. Кроме того, некоторые процедуры средствами алгоритмического языка могут записываться неэффективно. Эти причины обусловливают целесообразность составления отдельных блоков или целых программ с помощью языка низкого уровня типа мнемокода.

Системой математического обеспечения ЭВМ должны достигаться:

— максимально возможная автоматизация хода решения задачи с целью уменьшения вероятности появления ошибок, допущенных оператором;
— организация решения задачи в максимально возможном числе аварийных ситуаций для обеспечения безусловного решения задачи в заданные сроки;
— простота составления и отладки программ;

— эффективный контроль сохранности входной, промежуточной и выходной информации;
— контроль за работой ЭВМ и отдельных ее устройств. Основные виды контроля: логический контроль входной и выходной информации, логический контроль промежуточных результатов, контроль сохранности информации, контроль обмена информацией;
— возможность эффективного решения задач, входящих в АСУП;
— возможность простой сегментации программ, обеспечиваемой модульной структурой создаваемых программ.

Глава 2

ОПЕРАЦИОННАЯ СИСТЕМА

2.1. СТРУКТУРА ОПЕРАЦИОННОЙ СИСТЕМЫ ЭВМ «МИНСК-32»

Операционная система ЭВМ «Минск-32» состоит из комплекса программ, предназначенных для организации работы вычислительного оборудования, прохождения потока задач в вычислительной системе, связи оператора с машиной. Кроме того, в операционную систему (ОС) включены программные средства авто-

Рис. 2.1. Функциональный состав ОС ЭВМ «Минск-32»

мацизации программирования, отладки программ и формирования заданий для выполнения.

ОС состоит из двух крупных функционально различных частей: управляющей и обрабатывающей (рис. 2.1).

Управляющая часть ОС носит название (в терминологии документации по системе математического обеспечения ЭВМ «Минск-32») системы программ ДИСПЕТЧЕР.

Комплекс программ ДИСПЕТЧЕР является основной организующей частью операционной системы. Под управлением ДИСПЕТЧЕРа выполняются все обрабатывающие программы операционной системы и программы пользователя. К системе программ ДИСПЕТЧЕР относятся: НАЧАЛЬНЫЙ ВЫЗОВ, РЕЗИДЕНТ, КООРДИНАТОР, СОВМЕСТИМОСТЬ, РАЗГРУЗКА ЛЕНТЫ ВЫВОДА.

Программа НАЧАЛЬНЫЙ ВЫЗОВ выполняет функции загрузки в резидентную область памяти (РЗП) программ РЕЗИДЕНТ, КООРДИНАТОР и в управляющую область памяти (УОП) — первоначальных значений таблицы состояний внешних устройств машины, календарной даты и времени разгрузки системы. По окончании загрузки вышеуказанных программ программа НАЧАЛЬНЫЙ ВЫЗОВ автоматически включает в работу электронный датчик времени (ТАЙМЕР).

Программа РЕЗИДЕНТ контролирует сохранность в оперативной памяти (ОП) программ СОВМЕСТИМОСТЬ и КООРДИНАТОР и в зависимости от режима работы программы (в режиме совместимости с ЭВМ «Минск-22» или в режиме ЭВМ «Минск-32») загружает в ОП либо программу СОВМЕСТИМОСТЬ, либо программу КООРДИНАТОР. Одна из программ (либо СОВМЕСТИМОСТЬ, либо КООРДИНАТОР) и программа РЕЗИДЕНТ всегда находятся в ОП.

Программа КООРДИНАТОР выполняет в вычислительной системе следующие функции: обработку всех типов прерываний и выход на системные программы для обслуживания прерываний; планирование работы операций ввода-вывода и организацию их выполнения; организацию выполнения задач, включая динамическое распределение ресурсов вычислительной системы, динамический вызов программ в ОП, синхронизацию выполнения программ с операциями ввода-вывода, завершение работы программ при нормальном и ненормальном их исполнении; организацию совместного выполнения нескольких задач (мультипрограммный режим) с учетом их приоритета и готовности использовать время процессора; работу с таймером; устранение конфликтных ситуаций между программами.

Система программ КООРДИНАТОР состоит из программ: МОНИТОР, СУПЕРВИЗОР, АДМИНИСТРАТОР, АНАЛИЗАТОР СБОЕВ ПРОЦЕССОРА.

Программа МОНИТОР обеспечивает двухстороннюю связь оператора с КООРДИНАТОРом (или ЭВМ), прием заданий на

выполнение работ, подготовку и запуск рабочих программ на выполнение, формирует очередь заданий и обеспечивает включение программ в работу из очереди заданий.

Программа СУПЕРВИЗОР распределяет устройства за программами, запускает их в работу, реагирует на конец работы внешних устройств по приему-передаче данных, на их сбои, осуществляет связь с оператором при сбоях оборудования и устраняет последствия сбоев либо самостоятельно, либо по указаниям оператора.

Программа АДМИНИСТРАТОР обеспечивает синхронизацию работы программ КООРДИНАТОРа и рабочей программы, ведет учет времени и передает его значение исполняемой программе, включает программы в работу через определенные промежутки времени и по запросам внешних устройств.

При прерываниях по причине сбоев в работе внешних устройств или процессора СУПЕРВИЗОР передает управление программе АНАЛИЗАТОР СБОЕВ ПРОЦЕССОРА. При этом выполняются следующие действия: анализируется причина сбоев; если причина сбоя позволяет повторить процесс, то осуществляется его автоматическое повторение; сообщается оператору о наличии сбоя в вычислительной системе; принимается решение о целесообразности и возможности дальнейшего продолжения работы программы; осуществляется передача управления программе АДМИНИСТРАТОР для переключения на продолжение прерванной программы.

Выполнение рабочей программы может быть прервано и по директиве, указанию оператора через программу МОНИТОР.

Программа СОВМЕСТИМОСТЬ — это система программ, обеспечивающая выполнение на ЭВМ «Минск-32» программы, составленной на языке ЭВМ «Минск-22». Она выполняет следующие функции: ввод и подготовку к выполнению программы на языке ЭВМ «Минск-22», запуск такой программы, программную реализацию системы прерываний ЭВМ «Минск-22», обработку сбойных ситуаций ЭВМ «Минск-32» при выполнении программ ЭВМ «Минск-22», обработку прерываний по концу работы внешних устройств, учет времени работы программы, связь оператор-машина по управлению выполнением программы.

Систему программ СОВМЕСТИМОСТЬ составляют программы МОНИТОР, ИНТЕРПРЕТАТОР, АНАЛИЗАТОР СБОЕВ.

Программа МОНИТОР обеспечивает подготовку к выполнению очередной программы ЭВМ «Минск-22», переход от одной программы к другой и обеспечивает связь оператора с машиной.

Программа ИНТЕРПРЕТАТОР обеспечивает программную реализацию команд ЭВМ «Минск-22» на ЭВМ «Минск-32» и обслуживает систему прерываний.

Программа АНАЛИЗАТОР СБОЕВ обслуживает сбои процессора и внешних устройств, выдачу на управляющую пишущую машинку оператора через программу МОНИТОР сообщений или

указаний о возникших сбоях, некоторые действия по устранению последствий сбойных ситуаций.

Обрабатывающая часть ОС представляется комплексом инструктивно-методических материалов и программ автоматизации программирования задач, отладки программ на языках разных уровней и обслуживающих программ.

Основу обрабатывающей части ОС ЭВМ «Минск-32» составляют система символического кодирования (ССК), транслирующая система с алгоритмического процедурно-ориентированного языка КОБОЛ, система программ логического ввода-вывода (программы обмена на логическом уровне) и библиотеки программ общего применения.

Основу системы обслуживающих программ составляют программы корректировки ленты системы, программа автоматической сборки обобщенных программ, дублирования информации на внешних устройствах, программы тестов, автоматизации отладки и т. п.

2.2. ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ЭВМ

В основной комплект ЭВМ «Минск-32» входят: процессор, состоящий из центрального пульта управления (ЦПУ), устройства центрального управления (ЦУ), арифметического устройства (АУ), устройства обмена (УО), магнитного оперативного запоминающего устройства (МОЗУ); устройство УМП-23К с пишущей машинкой (ПМ) «Консул-254»; устройство управления записью и чтением на магнитную ленту (УМЛ-67) с пятью лентопротяжными механизмами (НМЛ-67); устройство ввода с перфокарт (УВвК); устройство вывода на перфокарты (УВК); устройство ввода с перфоленты (УВвЛ) с механизмом FS-1500; два устройства вывода на перфоленту (УВЛ) с механизмами ПЛ-80; устройство управления алфавитно-цифровой печатью (УПч) с механизмом АЦПУ-128/2; электронный датчик времени (ЭДВ).

Состав ЭВМ «Минск-32» может быть расширен произвольным набором внешних устройств, алгоритм связи которых соответствует универсальному принципу связи внешних устройств этой машины [30]. В частности, имеется возможность увеличения емкости МОЗУ до 65 536 машинных слов (ячеек). Минимальная емкость МОЗУ, обеспечивающая работу ЭВМ, составляет 16 384 ячейки.

В машине циркулируют два информационных потока, объединяющих работу устройств процессора и двух каналов связи. Два вида информационных потоков определяются наличием в машине двух основных информационных единиц обмена: 37-разрядного машинного слова и 7-разрядного символа.

В ЭВМ «Минск-32» имеются селекторный (быстрый) и мультиплексный (медленный) каналы связи.

Основной информационной магистралью, связывающей все устройства между собой, является тракт 37-кодовых шин (КШЧ).

По этому тракту машинные слова передаются: из МОЗУ в АУ, ЦУ, УО; из АУ, ЦУ, УО в МОЗУ; из одних регистров АУ в другие.

Для обмена символьной информацией между различными регистрами процессора служат информационные тракты внутренних кодовых шин символа (КШС). Например, передача символа из АУ в МОЗУ осуществляется путем выдачи соответствующих разрядов регистра Р2 на шины КШС, с которых код символа принимается в регистр символа устройства обмена.

Процессор предназначен для организации автоматической работы по вводу информации с разных носителей, выполнения арифметических и логических операций над информацией, находящейся в МОЗУ, вывода информации на внешние носители, а также синхронизации работы устройств ЭВМ в период выполнения программ.

ЦУ производит координацию всех действий в машине, а точнее обеспечивает автоматическое выполнение и синхронизацию работы всех устройств при выполнении программы. ЦУ выполняет функции запуска и останова машины, дешифрации кодов операций, организации последовательности тактов на выполнение операции (команды), формирования адресов МОЗУ и связи с МОЗУ, связи с АУ, обслуживания сбоев процессора. Обслуживание заключается в записи полученных сигналов сбоев в специальный регистр указателей сбоев.

Могут быть следующие виды (причины) сбоев процессора: «Подпрограмма», «Переполнение», «Недействительный код», «Не цифра», «Защита», «Чет», «ПУР».

Сбой «Подпрограмма» возникает при обращении в МОЗУ к занятой подпрограмме. Подпрограмма называется занятой, если содержимое ее первой ячейки отлично от нуля. Если произошло обращение к занятой подпрограмме, то вырабатывается сбой «Подпрограмма» и осуществляется автоматическое переключение процессора на выполнение программы первого уровня (см. п. 2.3).

Сбой «Переполнение» может возникнуть при выполнении арифметических операций над числами с фиксированной и плавающей запятой и в результате выполнения операций над десятичными числами. «Переполнение» может быть предусмотрено программой. В этом случае после команд, которые могут вызвать переполнение, должна следовать команда перехода по переполнению, что эквивалентно нормальному выполнению операции. Если в программе не предусмотрен переход по переполнению, то при его возникновении производится прерывание выполнямой программы и управление передается программе первого уровня с запоминанием состояния прерванной программы в соответствующих ячейках ее уровня.

Сбой «Недействительный код» возникает при чтении незадействованного в процессоре кода операции (переключение уровня, модифицирование индексных ячеек и др.), при попытке изменить границы защиты программы из рабочей программы и при исполь-

зовании в рабочей программе команд модифицирования ячеек управляющей области памяти, предназначенных только для служебных программ системы ДИСПЕТЧЕР. Этот сбой возникает также при попытке блокировать прерывание в рабочих программах. При возникновении сбоя «Недействительный код» формируются адреса сбоя и процессор переходит к анализу следующей команды программы. Если эта команда не является командой перехода по сбою «Недействительный код», происходит прерывание рабочей программы и управление передается программе нулевого уровня с запоминанием состояния прерванной программы в соответствующих ячейках ее уровня.

Сбой «Не цифра» возникает в случае, если тетрады десятичных цифр, участвующих в качестве операндов арифметической операции, имеют нецифровые комбинации кодов. При возникновении такого сбоя операция выполняется до конца, затем выбирается и анализируется следующая команда. Если эта команда не является командой перехода по сбою «Не цифра», происходит прерывание рабочей программы и управление передается программе нулевого уровня с запоминанием состояния прерванной программы в соответствующих ячейках ее уровня.

Сбой «Защита» возникает, если адрес передачи управления или адреса операндов выходят за границы оперативной памяти, отведенной для выполняемой программы. При этом сбое операция выполняется до конца без нарушения содержимого МОЗУ, не выделенного для программы, после чего программа прерывается и управление передается программе нулевого уровня с запоминанием состояния прерванной программы в соответствующих ячейках ее уровня.

Сбой «Чет» возникает в том случае, когда в прочитанной из МОЗУ ячейке или символе число двоичных единиц оказалось четным. В этом случае программа прерывается и управление передается программе нулевого уровня, если сбой «Чет» возник в тактах ЦУ, и программе первого уровня во всех других. Переключение на другой уровень осуществляется после запоминания состояния прерванной программы в соответствующих ячейках ее уровня.

Сбой «ПУР» (переключение уровня) возникает при любом вышеперечисленном сбое процессора во время переключения процессора с программы одного уровня на другой. При возникновении этого сбоя переключение осуществляется до конца, и лишь после этого анализируется следующая команда. Если это не команда перехода по сбою «ПУР», то происходит переключение процессора с выполняемой программы на программу нулевого уровня.

ЦУ выполняет также функции запоминания и ведения значений следующих регистров: текущего адреса выполняемой команды и увеличения его значения на единицу по окончании выполнения команды (регистр СЧАК); адреса записи, операнда, перехо-

дов, а также адреса, подлежащего проверке на соответствие границам защиты памяти; регистра базиса индекса (РБИ) и номера индекса; номера уровня выполняемой программы; номера базиса программы; индикаторов вычислителя (блокировка округления, нормализации, прерывания, нуля, «Минск-22», карты).

Арифметическое устройство осуществляет выполнение операций над адресами и числами в двоичной и десятичной системах счислений. В его состав входят: сумматор (СМ), регистр 1 (Р1), регистр 2 (Р2) и местное управление Р1 и Р2.

МОЗУ представляет собой набор ферритовых кубов (до 4-х) емкостью по 16 384 38-разрядных слов и устройства чтения и регистрации информации. 38-й разряд каждого слова МОЗУ является контрольным; его значение дополняет сумму первых 37 разрядов до нечетного значения и формируется при записи слова в МОЗУ. Контрольный разряд используется при чтении из МОЗУ слова для проверки его сохранности по нечету. Если сумма разрядов читаемой ячейки четна, то вырабатывается сбой «Чет». МОЗУ является оперативной памятью (ОП) ЭВМ «Минск-32».

УО обеспечивает связь процессора с внешними устройствами, следит за процессом обмена информацией. Устройство управления обменом обеспечивает подключение 104 внешних устройств к одному мультиплексному каналу и 32 внешних устройств — к селекторному. Подключаемые к мультиплексному каналу внешние устройства делятся на четыре группы: основную и три дополнительные. К основной группе отнесены внешние устройства: УПч, УВвК, УВК, УВвЛ, два УВЛ, УМП. К этой группе мультиплексного канала есть возможность подключения еще одного внешнего устройства ввода-вывода. Остальные 96 устройств медленного канала подключаются через специальные групповые коммутаторы, каждый из которых обеспечивает подключение 32 внешних устройства.

Мультиплексный канал позволяет параллельно работать всем 104 внешним устройствам. Это обеспечивается тем, что каждое внешнее устройство медленного канала имеет свой восьмиразрядный регистр приема-выдачи символа.

Подключение внешних устройств к селекторному каналу блочное, т. е. имеется возможность подключения четырех блоков, каждый из которых имеет восемь номеров внешних устройств и единое управление ими. ЭВМ «Минск-32» поставляется с одним блоком управления, обеспечивающим подключение восьми НМЛ-67. Имеется возможность подключения к селекторному каналу еще трех устройств управления для обеспечения связи с 24 внешними устройствами типа НМЛ-67, барабанов, машин «Минск-32».

Параллельно с работой процессора и устройствами, подключенными к мультиплексному каналу, может работать лишь одно устройство селекторного канала.

Взаимодействие процессора с внешними устройствами можно разбить на два независимых процесса: командное взаимодействие и информационное.

Имеются два вида командного взаимодействия — определение процессором возможности организации обмена информацией с нужным программе внешним устройством и запуск внешнего устройства для выполнения конкретного обмена. Первый вид командного взаимодействия заключается в опросе состояния требуемого внешнего устройства. Состояние каждого внешнего устройства фиксируется набором указателей. Обязательными для всех внешних устройств являются указатели: «Готово», «Занято», «Сбой». Кроме этих, каждое конкретное устройство может иметь свои специфические указатели, дополняющие его состояние. Например, для УВК — «Замятие», «Пустой карман» и т. п. Второй вид командного взаимодействия заключается в сообщении внешнему устройству необходимого кода операции и обеспечении возможности начала обмена.

Информационное взаимодействие заключается в непосредственном обмене информацией между процессором и внешним устройством. После осуществления командного взаимодействия, результатом которого является запуск механизма внешнего устройства и определение участка МОЗУ для обмена (засылки управляющего слова канала в соответствующую ячейку управляющей области памяти МОЗУ), процессор переходит к выполнению следующей команды программы. Этой командой может быть или команда ожидания конца обмена, или любая другая. В первом случае процессор будет ожидать окончания обмена (если в активном состоянии находится одна программа и в очереди на выполнение нет программ). Во втором случае процессор продолжит выполнение программы.

Говорят, что программа находится в активном состоянии, если она загружена в МОЗУ и ждет выполнения. При работе ЭВМ в мультипрограммном режиме после запуска внешнего устройства системой программ ДИСПЕТЧЕР определяется активная программа. Если такая имеется, то ей передается управление для продолжения выполнения, если нет, то включается на выполнение очередная программа из очереди заданий, а если в очереди нет заданий, то процессор переходит в режим ожидания ввода-вывода.

Определение программы, которой передается управление, осуществляется на основании значения ее приоритета и имеющихся в данный момент ресурсов ЭВМ.

Окончание обмена может определяться как процессором, так и внешним устройством. Окончание обмена характеризуется тем, что внешнее устройство перестает вырабатывать запросы на получение или выдачу очередного символа, устанавливает указатель внешнего устройства «Занято» в нулевое состояние и вырабатывает сигнал «Конец работы». Сигнал «Конец работы» является при-

чиной переключения процессора на выполнение соответствующей программы четвертого уровня (при окончании работы мультиплексного канала) и третьего уровня при окончании работы селекторного канала.

Электронный датчик времени — это устройство процессора, обеспечивающее формирование и получение значения текущего времени. Он может быть использован для организации разветвлений программ в процессе их выполнения, организации контроля и связи работы внешних устройств в электронной вычислительной системе на базе ЭВМ «Минск-32», накопления статистических данных и оформления отчетности по использованию машины потребителем, для организации автоматического контроля поступления информации в ЭВМ и т. п.

2.3. СТРУКТУРА ОПЕРАТИВНОЙ ПАМЯТИ

Вся область ОП ЭВМ «Минск-32» распределена на три участка: управляющую область памяти (УОП), резидентную область памяти (РЗП), рабочую область памяти (РОП).

УОП — постоянно распределенная область памяти, содержащая основные характеристики исполняемых программ и состояния оборудования ЭВМ.

РЗП — область ОП, постоянно закрепленная под программы системы ДИСПЕТЧЕР.

РОП — область ОП, предназначенная для программ пользователя.

Емкость ОП ЭВМ «Минск-32» может меняться от 16К* до 64К. Емкость РОП соответственно тоже может меняться. Емкость РЗП также может меняться за счет емкости РОП. В общем случае емкость РОП ($V_{\text{РОП}}$) определяется по формуле.

$$V_{\text{РОП}} = V - (V_{\text{УОП}} + V_{\text{РЗП}}),$$

где V — вся емкость ОП ЭВМ; $V_{\text{УОП}}$ — емкость ОП, занимаемой УОП; $V_{\text{РЗП}}$ — емкость ОП, занимаемой РЗП.

УОП занимает в ОП 0÷377 ячейки, которые постоянно распределены следующим образом: а) 12 полей уровней программ по восемь ячеек в каждом (ячейки 0÷137); б) регистры управляющих слов каналов (ячейки 140÷377). Содержимое ячеек поля уровня программы составляет информация, необходимая для начального пуска программы или ее продолжения.

Распределение ячеек поля уровня программы показано в табл. 2.1. При сбое процессора по одной из причин, приведенных в табл. 2.2, в соответствующем разряде 5-й ячейки поля уровня программы устанавливается единица. При отсутствии сбоя значение разряда равно нулю.

---

* К=1024 ячейки.

Таблица 2.

№ ячейки в поле уровня	№ начального разряда в ячейке	№ конечного разряда в ячейке	Длина поля в ячейке бит	Назначение поля
1	2	3	4	5
0	0	4	5	Не используется
	5	20	16	Значение первого базиса
	21	36	16	Значение нулевого базиса
1	0	4	5	Не используется
	5	20	16	Значение третьего базиса
	21	36	16	Значение второго базиса
2	0	37	37	Не используется
3	0	4	5	»
	5	20	16	Адрес верхней границы защиты памяти для программы (с точностью до 512 ячеек)
	21	36	16	Адрес нижней границы защиты памяти для программы (с точностью до 512 ячеек)
4	0	20	21	Не используется
	21	36	16	Базисный адрес индексных ячеек программы (с точностью до 16 ячеек)
5	0	4	5	Не используются
	5	11	7	Указатели сбоев процессора (табл. 2.2.)
	12	12	1	Не используется
	13	19	7	Индикаторы процессора (табл. 2.3)
	20	20	1	Не используются
	21	36	16	Пусковой адрес программы, или значение счетчика адреса команд (СчАК)
6	0	36	37	Содержимое сумматора, запомненное при аппаратном или программном переключении данной программы на другой уровень
7	0	13	14	Не используются
	14	17	4	Номер уровня программы, с которого произошло аппаратное или программное прерывание на программу данного уровня
	18	36	19	Не используются

Если в программе встретились операции по запрещению некоторых действий процессора или изменению режима его работы, то соответствующие разряды 5-й ячейки поля уровня программы принимают значения, равные единице. В противном случае их значение равно нулю.

Таблица 2.2

№ разряда	Наименование указателя (причины) сбоя процессора
5	Подпрограмма
6	Переполнение
7	Недействительный код
8	Не цифра
9	Защита
10	Чет
11	ПУР

Таблица 2.3

№ разряда	Наименование индикатора режима работы процессора
13	Блокировка округления
14	Блокировка нормализации
15	Нуль
16	«Минск-22М»
17	Карта
18	Блокировка прерываний
19	«Минск-22»

Наличие 12 уровней программ обеспечивает то, что в ОП могут одновременно находиться и выполняться 12 программ. Выполнение их может осуществляться последовательно или чередованием. Переход с одного уровня на другой (с выполнения одной программы на другую) осуществляется либо программно по специальным командам переключения уровней, либо автоматически по сигналам прерывания программ.

Уровни программ с 0÷6 и 13-й являются уровнями служебных программ и не используются для программ пользователя. Распределение уровней по программам приведено в табл. 2.4.

Таким образом, число программ пользователя, одновременно находящихся в ОП и выполняемых, может быть не более четырех, этим программам соответствуют 7÷12-й уровни.

Содержимое ячеек управляющей области может изменяться только с помощью команд изменения содержимого ячеек уровней,

а также с помощью команд ввода-вывода, переключений уровней, входа в программу и выхода из нее.

РОП предназначена для программ пользователя (программ, работающих на уровнях 7÷12). В силу того, что одновременно может выполняться более одной программы, каждой программе в ОП отводится своя область. Этим ограничивается вмешательство одних программ в области, отведенные для других. Если такое вмешательство происходит, то указатель «Защита» принимает значение единицы, что влечет за собой прерывание работы программы по причине «Защита».

Таблица 2.4

№ уровня	Наименование программы
0	Анализ сбоев процессора
1	Анализ сбоев каналов связи
2	Анализ сбоев внешних устройств
3	Обслуживание активных внешних устройств, устанавливающих связь с машиной, независимо от программ
4	Обслуживание внешних устройств, закончивших свою работу
5	Запуск внешних устройств
6	Обработка экстракодов
7	Программа пользователя
10	То же
11	» »
12	» »
13	Постоянной загрузки

В общем случае программа решения некоторой задачи состоит (с точки зрения распределения оперативной памяти) из поля команд (текст программы), полей обмена информацией между ОП и внешними устройствами (буферов ввода-вывода) и индексного поля (ИП). Из-за наличия возможности блочного построения программ и возможности мультиобработки возникает проблема сохранения содержимого используемых индексных ячеек при переходе от одной программы к другой. Для разрешения этой проблемы в операционной системе ЭВМ «Минск-32» принято соглашение о выделении в РОП поля индексных ячеек. Каждая программа имеет свое ИП, длина которого всегда кратна 16.

При сборке сложной программы, состоящей из нескольких автономно подготовленных программ, целесообразно объединить, если это возможно, поля обмена и, кроме того, в ряде случаев необходимо иметь общую для всех программ область, предназначенную для передачи некоторых параметров. В этом случае

появляется еще одно поле РОП — общая область. Таким образом, общая область РОП — это такая область памяти, которая доступна нескольким автономно подготовленным программам, выполняемым как одна программа. Наличие полей обмена общих областей и поля индексов в программе не обязательно.

Поля команд программ, входящих в сложную программу, размещаются последовательно друг за другом. Положение полей обмена каждой программы определяется при сборке сложной программы, причем для независимых программ выделяется одно и то же поле обмена. Длины общей области и поля команд определяются при сборке сложной программы, а индексного и обмена — как при сборке, так и при исполнении.

Для выполнения программы решения задачи выделяется один сплошной участок ОП, в начале которого размещается индексное поле, а в конце — поле команд.

При загрузке очередной программы индексное поле сложной программы увеличивается на длину индексного поля этой программы. При исключении такой программы из ОП длина индексного поля сложной программы уменьшается на величину индексного поля исключаемой программы.

2.4. ОБЩИЕ ВОПРОСЫ ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ ПРОГРАММ ДИСПЕТЧЕР

В минимальный комплект машины, необходимый для нормального функционирования системы программ ДИСПЕТЧЕР, входят: процессор с ОП емкостью не менее 16 384 ячеек; устройство УМП-23К с ПМ «КОНСУЛ-254»; УМЛ-67 с одним НМЛ-67 для магнитной ленты системы (ЛС), содержащей по крайней мере таблицу внешних устройств машины и программы РЕЗИДЕНТ, КООРДИНАТОР, СОВМЕСТИМОСТЬ, и с еще одним НМЛ-67 для магнитной ленты вывода (ЛЫ). На ленте вывода будут накапливаться результаты, которые необходимо выводить на УПч или УВК, а эти устройства заняты другими программами при работе ЭВМ в многопрограммном режиме. После освобождения требуемого устройства информация, накопленная на ЛЫ, может быть выведена программой РАЗГРУЗКА ЛЕНТЫ ВЫВОДА.

Каждая рабочая программа должна сообщить ДИСПЕТЧЕРУ перечень устройств, необходимых для ее выполнения. В задании на выполнение программы указывается требуемая ОП и требуемое количество накопителей на магнитной ленте (НМЛ). В случае выполнения программ в режиме совместимости, кроме того, ДИСПЕТЧЕРу задается таблица, в которой устанавливается соответствие между лентопротяжными механизмами ЭВМ «Минск-32» и ЭВМ «Минск-22».

Программам КООРДИНАТОР и СОВМЕСТИМОСТЬ необходимы данные о составе внешних устройств ЭВМ. Эти данные находятся в таблице внешних устройств (ТВнУ).

ТВнУ составляется для каждого конкретного образца машины и записывается на ЛС. При подключении к машине дополнительных внешних устройств или выходе из строя какого-нибудь из устройств основного комплекта на длительное время значения ТВнУ должны корректироваться.

ТВнУ состоит из двух частей. Первая часть этой таблицы содержит информацию в 32 строках, в первой из которых указывается количество подключенных к машине внешних устройств. В каждой последующей строке этой части ТВнУ приводятся необходимые сведения о конкретном типе внешнего устройства.

Вторая часть ТВнУ состоит из строк, в каждой из которых, кроме последней, находятся необходимые сведения о машинном номере подключенного к машине конкретного устройства и признак, указывающий, исключено это устройство или им можно пользоваться. В последней строке второй части ТВнУ указывается емкость РОП и РЗП в листах (листу соответствует 512 ячеек). Вторая часть ТВнУ может максимально состоять из 137 строк — одна строка на одно устройство. Если к машине подключены не все 136 ВнУ, то соответствующее количество строк будет нулевым.

Для настройки системы программ ДИСПЕТЧЕР в соответствии с комплектностью каждого конкретного образца машины должна быть сформирована ЛС. На ней, кроме программ ОС, обязательно должна быть записана ТВнУ, отражающая комплектность этого образца машины.

ТВнУ формируется в соответствии со списком оборудования, который составляется для каждого образца машины (табл. 2.5). Каждая строка этого списка содержит название устройства; условное обозначение типа ВнУ; цифровое обозначение устройства, состоящее из двух цифр, первая из которых является четверичной, а вторая — восьмеричной; машинный номер ВнУ, состоящий из трех восьмеричных цифр. В строке, характеризующей МОЗУ, вместо машинного номера ВнУ указывается количество листов МОЗУ в данном комплекте ЭВМ (этот параметр может равняться 40, 100, 140, 200), вместо цифрового обозначения типа ВнУ — количество листов МОЗУ, занимаемое РЗП. Эти данные записываются специальной обслуживающей программой «Формирование ленты системы» в начале ЛС.

При подключении к машине новых ВнУ или увеличении объема ОП необходимо скорректировать соответствующие значения ТВнУ. При отключении оборудования на длительный период времени также производится корректировка ТВнУ. При увеличении объема ОП необходимо корректировать первую строку списка. При каждой корректировке списка оборудования машины требуется повторное формирование ЛС.

В процессе выполнения программ под управлением программы КООРДИНАТОР или СОВМЕСТИМОСТЬ оператор может временно исключить любое ВнУ из числа используемых программами, а также разрешить использование ранее исключенных внеш-

Таблица 2.5

Название ВнУ	Условное обозначение типа ВнУ	Цифровое обозначение типа ВнУ	Машинный номер ВнУ
Магнитное оперативное запоминающее устройство	МОЗУ	15	100
Пишущая машинка пульта оператора	ПО	01	046
НМЛ с магнитной лентой системы	ЛС	02	000
НМЛ с магнитной лентой вывода	ЛЫ	03	001
НМЛ используемый в рабочей программе	МЛ	04	002
То же	МЛ	04	003
»	МЛ	04	004
Устройство ввода с перфолент	ВЛ	20	043
Устройство ввода с перфокарт	ВК	21	041
Устройство вывода на перфоленты	ЫЛ	30	040
То же	ЫЛ	30	042
Устройство вывода на перфокарты	ЫК	31	044
Устройство вывода на печать	ПЧ	32	045

них устройств, при этом состояние ТВнУ на ЛС сохраняется, а в ТВнУ, расположенной в ОП, производится лишь соответствующая отметка. Это делается оператором с помощью специальных директив.

Выполнение любой рабочей программы может производиться только под управлением системы программ ДИСПЕТЧЕР, точнее под управлением или программы КООРДИНАТОР, или программы СОВМЕСТИМОСТЬ. При выполнении очередного задания оператор имеет возможность продолжать ввод новых заданий или задавать МОНИТОРу другие директивы. МОНИТОР ставит очередные задания в очередь или включает на выполнение. АДМИНИСТРАТОР обеспечивает одновременное выполнение нескольких программ ЭВМ «Минск-32». Если оператор дал специальное указание на переход к работе в режиме совместимости, а в этот момент еще выполняются программы «Минск-32», то МОНИТОР запоминает это указание, не включает в работу новые задания на выполнение программ «Минск-32» и завершает выполнение активных программ. Далее производится вызов программы СОВМЕСТИМОСТЬ (через РЕЗИДЕНТ) и проверяется, имеется ли в очереди хотя бы одно задание на выполнение программ машины типа «Минск-22». Если нет, то организуется продолжение выполнения других программ «Минск-32».

После загрузки в ОП программ СОВМЕСТИМОСТЬ производится включение в работу программы машины типа «Минск-22»,

стоящей первой в очереди заданий. Программы СОВМЕСТИМОСТЬ последовательно выполняют находящиеся в очереди программы этого типа до возникновения одной из ситуаций:

а) в очереди нет больше заданий на выполнение программ машины типа «Минск-22»;

б) в очереди имеется срочное задание на выполнение программы «Минск-32»;

в) было указание оператора на переход к выполнению программ «Минск-32».

При возникновении какой-либо из этих ситуаций управление после окончания выполняющейся программы передается программе РЕЗИДЕНТ, которая вызывает в ОП программы КООРДИНАТОР для обеспечения выполнения программ «Минск-32».

Программа АДМИНИСТРАТОР управляет одновременным выполнением нескольких программ «Минск-32», при этом может оказаться, что для одной или нескольких программ нет свободных УВК и (или) УПч. Тогда результаты, выводимые на эти устройства, будут накапливаться на ЛЫ. При недостатке других ВнУ, за исключением лентопротяжных механизмов, включенная в работу программа не будет выполняться до освобождения требуемого ВнУ.

Информация, накопленная на ЛЫ, выводится по программе РАЗГРУЗКА ЛЕНТЫ ВЫВОДА на соответствующее устройство, причем в требуемой форме (как и при непосредственном выводе на ранее занятое устройство вывода).

В процессе выполнения рабочих программ у оператора могут возникнуть сомнения в правильности хранения в ОП программ ДИСПЕТЧЕР (например, из-за частых сбоев). В таких случаях оператор имеет возможность передать управление программе РЕЗИДЕНТ для осуществления соответствующей проверки. При неправильном хранении в ОП программы РЕЗИДЕНТ оператору выдается сообщение о необходимости заново выполнить программу НАЧАЛЬНЫЙ ВЫЗОВ, после чего в ОП не сохраняется очередь заданий. Оператору после выполнения программы НАЧАЛЬНЫЙ ВЫЗОВ нужно будет повторить ввод в ОП всех заданий, выполнявшихся или находящихся в очереди к моменту повторного выполнения программы НАЧАЛЬНЫЙ ВЫЗОВ. При неправильном хранении в ОП программ КООРДИНАТОР или СОВМЕСТИМОСТЬ программа РЕЗИДЕНТ организует повторный вызов нужной из этих программ с ЛС. Очередь заданий на выполнение программ при этом сохраняется.

Программы машины типа «Минск-22», выполняемые на машине «Минск-32» под управлением программ СОВМЕСТИМОСТЬ, могут иметь произвольную структуру, но они должны быть составлены в соответствии с руководством по программированию для машины типа «Минск-22». Не могут быть выполнены на машине «Минск-32» лишь те программы, в которых одновременная работа основной и прерывающих программ основана на исполь-

зовании временных характеристик машины типа «Минск-22». Это возможно, например, в случаях, когда результаты, получаемые в основной программе, выводятся в прерывающей, работающей одновременно с этой же основной, в которой на месте выводимых в режиме прерывания результатов формируются новые результаты. Так как машина «Минск-32» имеет значительные отклонения во временных характеристиках по сравнению с машиной типа «Минск-22», то такие программы на машине «Минск-32» могут быть выполнены неправильно.

Отдельно рассмотрим требования к программам, предназначенным для одновременной работы в многопрограммном режиме.

Для выполнения программы на машине предварительно должно быть составлено задание. По одному заданию может быть выполнена одна программа или пакет. В задании отражаются следующие сведения о программе: имя, примерное время выполнения, требуемый объем ОП в листах, требуемое количество НМЛ.

Задание подготавливается на одном из машинных носителей информации и по специальной директиве передается оператором программе МОНИТОР, которая ставит задание в очередь на выполнение.

Возможны следующие случаи включения в работу задания из очереди: сразу после постановки его в очередь, если в момент приема задания в очереди не было других заданий, а МОНИТОР должен был включать в работу новое задание; после окончания очередного задания, когда поставленное в очередь задание является срочным (с высоким приоритетом) или в очереди нет больше заданий.

Перед включением в работу первой программы задания проверяется, достаточно ли свободных ресурсов вычислительной системы для выполнения этой программы. При отсутствии достаточного объема ОП и количества НМЛ эта программа не включается в работу. Если в процессе выполнения программы осуществляется обращение к УВЛ или ПМ и если эти устройства заняты, то выполнение этой программы прерывается до освобождения соответствующего устройства. Если же производится обращение к УВК или УПч, то при отсутствии свободных соответствующих устройств информация, выводимая на эти устройства, накапливается на ЛЫ.

При включении задания в работу печатается сообщение о начале его выполнения и управление передается к загрузке в ОП первой программы. Вызов программы в ОП и ее настройка по заданному месту ОП производится ЗАГРУЗЧИКом — специальным блоком программы КООРДИНАТОР. Загрузка заключается во вводе программы в ОП с внешнего носителя и преобразовании некоторых объектов программы, зависящих от ее месторасположения в ОП. По требованию к месторасположению о ОП программы подразделяются на перемещаемые и неперемещаемые.

Перемещаемые — это программы, которые могут выполняться после размещения с любого свободного адреса в ОП. Неперемещаемые — это программы, которые должны выполняться после размещения в ОП всегда с одного и того же адреса.

Для того чтобы программа могла быть обработана ЗАГРУЗЧИКом, она должна быть записана на его языке. Программа на языке ЗАГРУЗЧИКа, кроме команд собственно программы, содержит дополнительную информацию о правилах размещения ее объектов в ОП и некоторые сведения о других программах, используемых в ней.

Включенная в работу программа начинает выполняться в соответствии с присвоенным ей приоритетом, независимо от других программ, выполняемых одновременно с ней. Прерывание выполнения программы и переход к программе КООРДИНАТОР осуществляется только при появлении одной из следующих ситуаций: встретился один из операторов связи программы с КООРДИНАТОРом, встретился один из операторов обращения к внешнему устройству, закончило работу одно из внешних устройств, оператор подал запрос на ввод директивы, появился сбой процессора или ВнУ в момент выполнения программы, закончился определенный интервал времени.

Операторы связи программы с КООРДИНАТОРом определяют те действия, которые должен выполнить КООРДИНАТОР для нормального продолжения программы. К этим действиям относятся: закрепление за программой требуемого ВнУ и освобождение его данной программой; требование на ожидание окончания работы ВнУ; требование на выдачу даты, времени и управляющего слова канала для заданного ВнУ; запоминание адреса программы, с которого нужно повторить программу при возникновении сбойной ситуации; загрузка программы; окончание работы программы и т. п.

При появлении одной из этих ситуаций выполнение программы прерывается, а управление передается программе КООРДИНАТОР, где ситуация расшифровывается и производится соответствующее действие. После этого управление передается координирующему блоку, который переходит к выполнению одной из одновременно выполняемых программ.

По операторам связи с внешними устройствами СУПЕРВИЗОР производит либо обращение к внешнему устройству для выполнения требуемой работы, либо осуществляет опрос состояния требуемого ВнУ.

При возникновении сбойной ситуации в процессе выполнения программы производится либо повторное обращение к внешнему устройству (при сбоях ВнУ), либо повторение программы с адреса, установленного ранее в операторах связи программы с КООРДИНАТОРом.

По истечении заданного интервала времени также производится прерывание выполняемой программы. В этом случае КООР-

ДИНАТОР выдает на пишущую машинку сообщение об истечении времени и переходит к координирующему блоку, где управление передается другой программе.

Внешнее устройство, к которому было произведено обращение из программы, после окончания заданной работы также вызывает прерывание, и управление передается СУПЕРВИЗОРу, в котором осуществляются проверки правильности окончания заданной работы и наличия в ОП программы, которая ждет окончания работы этого ВнУ. При нахождении такой программы разрешается продолжение ее выполнения.

При выполнении программы машины типа «Минск-22» сохраняются в принципе те же ситуации, возникновение которых приводит к прерыванию выполнения рабочей программы и переходу к программе СОВМЕСТИМОСТЬ. Исключением является отсутствие специальных операторов связи и обращения к внешним устройствам. Роль таких операторов играют специфические команды машины типа «Минск-22». При появлении одного из таких операторов управление передается программе СОВМЕСТИМОСТЬ, где он расшифровывается и производится его программная реализация. После этого управление передается на продолжение рабочей программы. По окончании работы ВнУ возможны два варианта: продолжение выполнения рабочей программы или программная реализация процесса прерывания машины типа «Минск-22».

Взаимодействие оператора с программами системы ДИСПЕТЧЕР и рабочими программами осуществляется через пишущую машинку пульта оператора. Для этой цели служат директивы, передаваемые оператором ДИСПЕТЧЕРу, и сообщения или указания, выдаваемые ДИСПЕТЧЕРом или рабочими программами оператору.

Под сообщением понимаются сведения, информирующие оператора о состоянии машины и о действиях, выполненных ДИСПЕТЧЕРом или рабочими программами. После выдачи сообщения соответствующая программа продолжает свою работу и не требует никакого вмешательства.

Под указанием понимается информация о необходимости принятия решения и выполнения определенных действий. Приняв решение и выполнив эти действия, оператор должен (в соответствии с инструкциями по выполнению программ) специальной директивой передать ответ о принятом решении и выполненных действиях. Приняв этот ответ, ДИСПЕТЧЕР осуществляет продолжение прерванной программы или переходит к выполнению других программ.

В процессе работы ЭВМ оператор может вызывать в ОП отдельные программы системы ДИСПЕТЧЕР, контролировать правильность их хранения в ОП, передавать ДИСПЕТЧЕРу задания на выполнение программ машин «Минск-32» и типа «Минск-22», исключать из очереди задания, запрещать на некоторое время

выполнение отдельных заданий, продолжать выполнение ранее остановленных программ с заданного места, исключать из работы и включать в работу ВнУ из числа подключенных к машине и описанных в ТВнУ, передавать ДИСПЕТЧЕРу сведения об истинном времени и отвечать о принятом решении и выполненных действиях в соответствии с полученным от ДИСПЕТЧЕРа указанием, передавать ряд других специфических директив по управлению выполнением программ.

2.5. ЛОГИЧЕСКАЯ ОРГАНИЗАЦИЯ ДАННЫХ

Логической называется такая организация данных, как она представляется программисту вне зависимости от физического носителя, на котором расположены эти данные. Минимально измеримая логическая единица данных — логическая запись. На содержание логической записи не накладывается никаких ограничений, и программист вправе любую часть своих данных представить как логическую запись. Логической записью может быть анкета на одного работающего, блок некоторой программы, данные о состоянии некоторой единицы оборудования, потребность в некотором материале для изготовления ряда деталей и т. п.

Более крупной организационной логической структурой данных является массив, состоящий из одной и более логических записей.

При организации данных на внешних носителях, нам, как правило, не безразлично, как они будут располагаться на них. Информация о расположении данных на внешних носителях необходима для организации поиска, ввода в ОП и ее обработки. Существует три организационных способа формирования массивов из логических записей: последовательный, индексно-последовательный, прямой. При последовательном способе формирования массивов логические записи переносятся на машинный носитель в порядке их поступления и каждая следующая запись располагается за предыдущей. При индексно-последовательном способе месторасположение каждой следующей записи зависит от значения некоторого реквизита-ключа (индекса) в предыдущей и очередной логических записях. При прямом способе организации массивов месторасположение (адрес) каждой записи на машинном носителе определяется некоторой программой пользователя. Рассмотренные способы формирования массивов в общем случае не исключают, а дополняют друг друга при формировании массивов логических записей.

В массив могут записываться как единичные логические записи, так и сблокированные, когда при одном обращении к вводу записывается несколько логических записей. Количество логических записей, объединенных в один блок вывода, будем называть коэффициентом блокирования. Способ формирования массивов из логических записей при коэффициенте блокирования, большем

единицы, будем называть базисным. По аналогии со способами формирования массивов имеется четыре способа доступа к логическим записям в массиве: последовательный, индексно-последовательный, прямой и базисный.

Базисный способ доступа — это способ ввода логических записей на уровне блоков. Внутри блока логические записи могут быть организованы последовательным, индексно-последовательным или прямым способом.

При прямом способе доступа к данным месторасположение любой логической записи определяется программой пользователя при обращении к вводу.

При последовательном способе доступа выборка логических записей из массива производится в соответствии с последовательностью их расположения на носителе, т. е. одним обращением к вводу программа получает доступ к очередной логической записи.

При индексно-последовательном способе доступа производится выборка логических записей из массива данных последовательно, в соответствии с порядком ключей, хотя их расположение на носителе может быть совсем другим. Кроме того, по значению реквизитов-ключей можно обращаться за записями в произвольном порядке.

Сочетание способов формирования массивов и доступа к логическим записям в массивах носит название методов доступа к данным.

Способы доступа определяются в основном организационной структурой данных. Однако в зависимости от каких-то критериев (как правило, экономических) на практике приходится использовать способы доступа с каким-то отклонением от способов формирования массивов или чередовать разные способы доступа на одной организации массивов. Например, на этапе внутренней сортировки информации, находящейся на магнитных лентах, целесообразно использовать для ввода базисный способ доступа, для их упорядочения — прямой, а на этапе слияния упорядоченных логических записей — последовательный способ.

Программы доступа в терминологии документации по системе математического обеспечения ЭВМ «Минск-32» называются программами обмена. Такое название этого раздела программ, на наш взгляд, несколько неточно. В силу распространенности ЭВМ «Минск-32» и документации по системе математического обеспечения этой машины мы не везде будем изменять терминологию. Необходимо только помнить, что фразы «программы доступа» и «программы обмена» надо понимать однозначно.

Независимо от логической организации данные, записанные на носителе, всегда разбиваются на части, определяемые спецификой самого носителя и емкостью оперативной памяти. Организация данных, получающаяся вследствие такого деления, носит название физической организации. Максимальная физическая единица

информации носит название том. Примеры томов — колода перфокарт, катушка с перфолентой.

Информация, размещенная в одном томе, как правило, может быть разбита на более мелкие физические единицы. Например, информация на катушке магнитной ленты разбивается на зоны, отделяемые межзонными промежутками. Характерной особенностью физической записи в массиве является то, что за одно обращение к внешнему устройству читается или записывается одна физическая запись. Другие примеры физических записей — перфокарта, строка алфавитно-цифрового печатающего устройства.

Логическая и физическая организация данных могут не совпадать друг с другом. Например, на одном томе может быть расположено несколько массивов, несколько логических записей могут быть объединены в одной физической.

В одном массиве могут содержаться логические записи одного из трех форматов: фиксированного, переменного, неопределенного.

При фиксированном формате длины всех логических записей в массиве постоянны. Коэффициент блокирования логических записей в одной физической также постоянен для всего массива. Исключение может составлять только последняя физическая запись, которая может содержать неполное количество логических записей, и будет, следовательно, короче остальных. Для выделения логических записей программисту должны быть известны длина логических записей и коэффициент блокирования.

При переменном формате как длины логических записей, так и их количество в одной физической могут принимать на протяжении массива различные значения. Длина каждой логической записи указывается в ее первой ячейке, точно таким же образом указывается длина каждой физической записи. Благодаря этому при вводе записей переменного формата программа всегда располагает сведениями о длине каждой (логической и физической) записи в массиве. Для выделения буферов ввода-вывода программист должен знать максимально возможные длины физических и логических записей в обрабатываемом массиве.

При неопределенном формате длины записей также могут меняться на протяжении массива. Прямого указателя длины логических записей в массиве не имеется. Она определяется неявным образом через символ-разделитель.

Записи фиксированного и переменного форматов могут быть блокированными или нет. Таким образом, возможны следующие пять форматов записей:

ФН — записи фиксированной длины, неблокированные;
ФБ — записи фиксированной длины, блокированные;
ПН — записи переменной длины, неблокированные;
ПБ — записи переменной длины, блокированные;
НН — записи неопределенной длины.

Как было определено в разд. 2.3, поля рабочей области памяти, в которые данные вводятся для обработки и из которых выводятся после обработки, называются буферами ввода-вывода (БВВ). Для одного массива может быть несколько БВВ, а для разных массивов могут использоваться одни и те же БВВ. Совокупность буферов, выделяемых для некоторой программы, называется ее буферным фондом. Использование нескольких БВВ для одного массива и разных — для разных массивов является необходимым условием для организации совмещения обработки очередных логических записей с вводом-выводом последующих этого же массива.

Технологический маршрут прохождения информации в программах обработки данных в общем случае сводится к последовательности, приведенной на рис. 2.2.

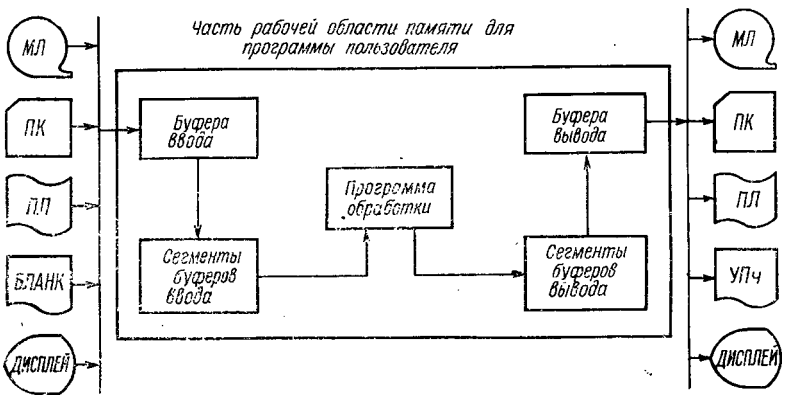


Рис. 2.2. Технологический маршрут прохождения информации при решении задач

По запросу программы с некоторого внешнего устройства в ОП вводятся данные. Эти данные размещаются в буфере ввода. Программа определяет один из режимов доступа к данным в буфере ввода-вывода: режим указания места или режим пересылки.

Режим указания места сводится к тому, что программы, реализующие доступ к данным, определяют абсолютное значение адреса очередной записи, с помощью которого в программе пользователя можно осуществлять обработку этой записи. При использовании режима пересылки очередная запись из буфера ввода пересылается в специальный сегмент буфера ввода, равный длине одной записи.

Результатные данные (логические записи) также можно размещать сначала в сегментах буферов вывода, а затем пересылать в буфер вывода или можно их сразу размещать в буфере вывода. Отсюда следует, что в конкретных программах пользователя

сегменты буферов ввода-вывода могут отсутствовать полностью или частично.

Прямой и индексно-последовательные методы доступа — это методы доступа к данным массивов, размещенных на внешних запоминающих устройствах с произвольным методом доступа (магнитные барабаны, магнитные диски и т. п.).

В силу того, что в основной комплект ЭВМ «Минск-32» не входят такие внешние устройства, основным методом доступа к данным для этой ЭВМ является последовательный.

Массивы и тома, помимо записей, составляющих их содержание, могут иметь дополнительные записи, называемые метками. В метках массивов и томов записываются признак метки, имя массива или тома и др. Эти данные предназначены для определения, тот ли массив (том) используется в задаче, для защиты данных от неавторизованного использования и т. п. В терминологии документации по системе математического обеспечения ЭВМ «Минск-32» метки называются контрольными блоками.

По назначению и месторасположению на носителе имеются следующие контрольные блоки:

НК — начало катушки МЛ, бобины перфоленты (ПЛ), колоды перфокарт (ПК);

КК — конец катушки, бобины ПЛ, колоды ПК;

НБ — начало массива;

КБ — конец массива.

Если массив не поместился в одном томе, то перед блоком КК записывается промежуточный контрольный (ПКБ), означающий, что данный массив имеет продолжение в другом томе, в новом томе продолжение массива после метки НК начинается с промежуточного начального блока (ПНБ). Если для каких-то целей не хватает поля для записи характеристик массивов, то в СМО ЭВМ «Минск-32» разрешается ведение дополнительного блока (ДБ). Такие блоки могут располагаться на носителе только за НБ или за КБ. Программы, реализующие методы доступа в СМО ЭВМ «Минск-32», никакой обработки и записи ДБ не производят. Формирование и обработку таких блоков должны осуществлять программы пользователя.

НК и КК состоят из записи длиной в две ячейки. В первой из них размещен признак метки (*** НК или *** КК), а во второй — имя катушки (тома). Для КК вторая ячейка может состоять из пробелов или признака метки.

НБ и ПНБ состоят из признака метки (*** НБ и ** ПНБ соответственно), имени массива, срока годности записанного массива, даты образования массива, типа обмена, указывающего, по пять или шесть символов в ячейке производится обмен, признака формата записей в массиве, указывающего, из записей какой длины — фиксированной, переменной или неопределенной — состоит массив. Если массив состоит из записей неопределенной длины, то указывается символ-разделитель записей. Для записей

фиксированного формата указывается длина записи. Кроме того, для ПНБ указывается порядковый номер тома.

КБ и ПКБ обязательно содержат признак блока и имя массива, кроме того, могут содержать также количественные характеристики массива (количество зон или записей на МЛ, количество перфокарт или символов на перфоленте).

В обрабатывающей части операционной системы ЭВМ «Минск-32» имеется библиотека программ обмена, включающая программы по подготовке массивов к доступу и по записи-чтению информации этих массивов.

Подготовка массивов к доступу зависит от того, будет информация записываться в массив или читаться из него. При подготовке массива к записи ищется конечный контрольный блок катушки МЛ и вместо него записывается НБ, после чего можно записывать информационные блоки этого массива.

При подготовке массива для чтения осуществляется поиск нужного массива на МЛ, для чего все зоны, начиная с первой после контрольного блока начала катушки, читаются с МЛ, и проверяется, равно ли количество символов в очередной зоне 80. Начальный и конечный контрольные блоки массива на МЛ занимают по 80 символов. В таких зонах проверяется содержимое первой ячейки на совпадение с признаком начального контрольного блока массива (НБ). И только после этого сравнивается имя искомого массива с содержимым второй ячейки НБ, где находится имя записанного на МЛ массива. Их совпадение означает, что нужный массив найден.

Новый массив на магнитной ленте может быть записан обычно только после тех массивов, которые уже имеются на этой катушке. Изменять информацию в массиве, за которым находится другой массив, и записывать новый на место старого массива нельзя, поскольку при записи нового массива не гарантируется сохранение информации последующих массивов.

Из-за особенностей подготовки информации на перфоленте контрольные блоки массивов могут содержать меньше 80 символов. Количество символов в начальных блоках массивов, подготовленных на перфоленте, строго не зафиксировано. Установлен разделительный символ *, служащий границей (последним символом) информации начального блока. Конечный контрольный блок массива на перфоленте всегда должен содержать только 20 символов.

На перфокартах используются только обязательные контрольные блоки (НБ и КБ), все остальные контрольные блоки (НК, КК, ПНБ, ПКБ и ДБ) запрещены. Каждый НБ и КБ занимает одну перфокарту. Количество перфокарт с информацией, заключенных между картами с блоками НБ и КБ, не ограничено.

Вся выводимая информация на УПч, оформленная как одно целое, называется печатным сообщением. Под оформлением печатного сообщения понимается печать титульного листа, заголов-

ка и концовки страниц, печать информационных строк сообщения в заданном порядке, нумерация листов сообщения.

По аналогии с массивами на других машинных носителях в массиве, выдаваемом на УПч, также можно различать контрольные блоки. Информация начального блока содержится в титульном листе, конечного — в тексте окончания последнего листа сообщений. Весь массив состоит из набора информационных блоков — печатных листов, пронумерованных по возрастанию, начиная с нулевого (титульного листа). В массиве, выводимом на печать, допускаются как промежуточные блоки (заголовок и окончание каждого листа), так и дополнительные (нестандартные заголовки и окончания). В отличие от массивов на других носителях информация контрольных блоков печатного сообщения произвольна как по содержанию, так и по размеру. Смысл и назначение контрольных блоков массивов, организованных на всех видах машинных носителей, одинаков.

2.6. ОРГАНИЗАЦИЯ ОБМЕНА ИНФОРМАЦИЕЙ

Обмен информацией между оперативной памятью машины и внешними устройствами основного комплекта реализуется программами обмена, выполняющими следующие функции: поиск, ввод и проверку контрольных блоков входных массивов, формирование и вывод контрольных блоков выходных массивов данных, ввод данных в буфер ввода, вывод данных из буфера вывода, объединение записей в буфера вывода из их сегментов, выделение логических записей из буферов ввода в сегменты ввода, контроль правильности обмена и организация выхода на обработку обнаруженных ошибок.

Исходной информацией для работы программ обмена являются: описание обмениваемых массивов; перечень адресов внешней программы, берущей на себя обработку ошибок; адреса в рабочей области памяти для буферов ввода-вывода и их сегментов. Вся необходимая информация для работы программ обмена сосредоточивается в соответствующих таблицах описаний. Содержание и структура этих таблиц стандартизованы. Это означает, что вне зависимости от применяемой программы обмена количество, содержание и правила заполнения таблиц описаний не меняются. Изменяются лишь значения операндов. Запись операндов и операторов (строк таблиц описаний) подчиняется правилам записи программ и данных в системе символического кодирования ЭВМ «Минск-32». Для работы программ обмена в общем случае необходимы описание массива (табл. 2.6) и описание буфера ввода-вывода (табл. 2.7).

Рассмотрим приведенное в табл. 2.6 описание массива. Д — этикетка (символический адрес) описания массива. Имя массива — наименование соответствующего массива, состоящее из пяти символов. Срок годности — год, месяц, число, определяющие по-

Таблица 2.6

Этикетка	КОП	Адреса и замечания
Д	КТ	<имя массива>
	НОП	
	КЧ	± <срок годности>
	НОП	
	КВНУ	<устройство>
	КЧ	<запись>
	НОП	
	НОП	

Таблица 2.7

Этикетка	КОП	Адреса и замечания
А	КА	АН; АК
	КА	В; С
	КА	ЗН; ЗК
	КА	ОФ; АС

следний срок возможности использования информации из массива, знак перед этим параметром определяет, по пять (+) или по шесть (−) символов в ячейке осуществляется обмен между оперативной и внешней памятью машины. Устройство — тип и номер устройства, на котором расположен массив. Запись — параметры, определяющие формат, длину или разделитель записи, признак вывода каждой записи отдельной зоной и длину буфера ввода-вывода. Вторая, четвертая и седьмая ячейки описания массива используются как рабочие. Во второй ячейке формируется информация о номере обрабатываемого тома (бобины) внешнего устройства. В четвертой ячейке находится номер очередной зоны, подлежащей обмену. Для массивов на перфокартах или перфолентах в этой ячейке организуется счетчик карт или символов; для массива, выводимого на УПч,— номер очередного печатного листа. В седьмой ячейке содержится абсолютный адрес очередной обмениваемой записи.

В общем случае для каждого обрабатываемого массива должно быть составлено свое описание.

В описании буфера ввода-вывода (табл. 2.7) АН (АК) — начальный (конечный) адрес буфера ввода-вывода (поля обмена). Если буфер ввода-вывода равен длине одной зоны, то АК=0. В — начальный адрес программы пользователя, обрабатывающей контрольные блоки при вводе массива или формирующей проме-

жуточные контрольные блоки при выводе массива. С — начальный адрес программы пользователя, обрабатывающей возможные сбойные ситуации. ЗН (ЗК) — начальный (конечный) адрес сегмента буфера ввода-вывода, предназначенного для размещения одной записи массива. Последняя ячейка описания буфера заполняется только для массива, выводимого на УПч, при этом ОФ — адрес описания оформления печатного листа, АС — адрес описания информационной строки.

В начале любого буфера ввода-вывода отводится три ячейки, используемые в качестве рабочих. Информация размещается с адреса АН+3 буфера. В ячейке АН+1 находится число, определяющее объем информации в соответствующем буфере. Это число увеличивается по мере ввода данных в буфер и уменьшается по мере вывода данных из буфера.

Обмен информацией с внешними устройствами машины реализуется программами логического доступа (обмена), которые можно разделить на следующие шесть групп: подготовка к обработке массивов данных, ввод данных в память машины с внешних носителей, вывод данных из памяти машины на внешние носители, перемещение записей в оперативной памяти машины, завершение обработки массивов данных, вспомогательные программы.

К первой группе относятся программы:

— открыть входной массив на магнитной ленте (ОТВМЛ);
— открыть входной массив на перфокартах и перфоленте (ОТВКЛ);
— открыть выходной массив на магнитной ленте (ОТЫМЛ);
— открыть выходной массив на перфокартах, перфолентах и печати (ОТЫ).

Программы этой группы производят необходимые приготовления к началу обмена данными с внешними устройствами. К этим действиям относятся: очистка рабочих ячеек описания массива, формирование в описании массива признака «массив открыт», поиск на магнитной ленте входного массива или места для записи выходного массива, ввод и проверка начальных контрольных блоков входных массивов или формирование и вывод начальных контрольных блоков выходных массивов (кроме выводимого на печать).

Ввод данных в память машины с внешних носителей осуществляется программами:

— ввести с магнитной ленты (ВМЛ);
— ввести с перфоленты (ВПЛ);
— ввести с перфокарт (ВПК).

Эти программы переносят информацию с внешних носителей в буфера ввода. Во время ввода производится соответствующий контроль, формируется и записывается информация о количестве ячеек, занятых в буфере ввода введенными данными. Ввод с магнитной ленты осуществляется как по одной зоне данных, так и несколькими зонами в зависимости от размера буфера.

При вводе перфокарт должна вводиться информация, определяющая число карт. В зависимости от буфера ввода число вводимых карт может меняться.

С перфоленты также можно вводить как запись, так и часть массива в зависимости от размера буфера ввода.

Вывод данных из памяти машины на внешние носители производится программами:

— вывести на магнитную ленту (ЫМЛ);
— вывести на перфоленту (ЫПЛ);
— вывести на перфокарты (ЫПК);
— вывести на печать (ЫПЧ).

Эти программы переносят информацию из буфера вывода на внешние носители. При выводе производится соответствующий контроль и корректируется количество информации, оставшейся в буфере (поле обмена). На магнитную ленту информацию из буфера вывода можно записать одной или несколькими зонами. В последнем случае все зоны должны быть постоянной длины (кроме последней).

На перфокарты данные выводятся либо записями, либо массивом карт из буфера вывода. В любом случае выводится целое число перфокарт. Вывод отдельных записей допускается как из сегмента буфера, так и из буфера.

На перфоленту, так же, как и на перфокарты, можно вывести либо отдельную запись (из сегмента буфера записи), либо массив символов из буфера.

На печать можно вывести как одну, так и несколько записей, состоящих из произвольных символов в кодах ГОСТ 10859—64. Каждая запись печатается отдельной строкой, независимо от того, содержится ли запись в сегменте буфера или в буфере. При печати ведется подсчет числа строк на одном листе. Переход к новому листу, оформление заголовка и концовки листов по заданному шаблону, а также нумерация печатных листов производятся автоматически. Кроме этого, если позволяет ширина печатного листа, несколько листов могут печататься параллельно.

Перемещение записей в оперативной памяти машины осуществляется программами:

— прочитать запись (ЧТЗ);
— записать запись (ЗПЗ).

Программа чтения записи переносит очередную запись из буфера ввода в сегмент буфера. Записи фиксированного, переменного и неопределенного формата помещаются с нулевого символа первой ячейки сегмента буфера. Независимо от формата записи, в контрольной ячейке сегмента буфера формируется информация о длине записи. Если очередная запись не вмещается в сегменте буфера, то ее обработка должна осуществляться специальной программой пользователя, начальный адрес которой (В) указывается в описании буфера ввода-вывода.

Если при обращении к программе чтения записи не задается адрес сегмента буфера, то в седьмой ячейке описания массива сформируется адрес начала очередной записи в буфере ввода, т. е. будет применен режим указания места. По мере исчерпания информации в буфере ввода производится автоматическое обращение к одной из программ ввода данных в память машины с внешних носителей.

Программа ЗПЗ переносит в буфер вывода содержимое сегмента вывода. По мере заполнения буфера вывода производится автоматическое обращение к одной из программ вывода данных из памяти машины на внешние носители.

К программам завершения обработки массивов относятся:
— закрыть входной массив (ЗАВ);
— закрыть выходной массив (ЗАЫ).

Эти программы производят действия, необходимые в конце обработки массивов: гашение признаков обработки в описании массива, ввод и проверку конечных контрольных блоков входных массивов или формирование и вывод конечных контрольных блоков выходных массивов и др.

К вспомогательным программам относятся:
— сформировать срок годности (СРОК);
— ввести контрольный блок (ВБЛОК);
— вывести контрольный блок (ЫБЛОК);
— оформить сообщение, выводимое на печать (ОФОРМ).

Программа СРОК используется для формирования в описании массива срока годности.

Программа ВБЛОК вводит с исходного состояния магнитной ленты в память машины контрольный блок с соответствующей проверкой номера блока и его контрольной суммы.

Программа ЫБЛОК формирует в буфере вывода контрольный блок и выводит его на магнитную ленту.

Программа ОФОРМ предназначена для вывода в заданном порядке оформительского текста (титульного листа, концовок и заголовков страниц). Она используется в совокупности с программой ОТЫ (открыть выходной массив), а также вместо программы завершения печати.

Обращение к программам, реализующим методы логического доступа к данным, осуществляется по общим правилам обращения к внутренним программам на языке символического кодирования ЭВМ «Минск-32».

Глава 3

ТИПОВЫЕ ПРОЦЕДУРЫ ОБРАБОТКИ ЭКОНОМИЧЕСКОЙ ИНФОРМАЦИИ

3.1. ОБЩИЕ СВЕДЕНИЯ

Задачи, решаемые в автоматизированных системах управления предприятием, относятся к классу информационно-экономических. Можно выделить следующие особенности задач этого класса:

— большой объем исходных, промежуточных и выходных данных;
— сложная информационная взаимосвязь задач, выражающаяся в использовании одних и тех же массивов или подмассивов информации в разных задачах;
— наличие больших объемов накапливаемой информации с достаточно длительным сроком хранения;
— на данном этапе развития вычислительной техники массивы информации размещаются на магнитных лентах (МЛ), причем каждый массив может неоднократно просматриваться в процессе решения задачи;
— большое число массивов, участвующих в решении задач;
— блочное построение программ обработки информации, причем программа каждой задачи разбивается на сравнительно большое число этапов;
— зависимость хода решения задачи от значения некоторых параметров, получаемых в ходе решения;
— необходимость сегментации массивов информации, расположенных на МЛ, связанная со сравнительно малым объемом оперативной памяти ЭВМ;
— достаточно жесткая регламентация времени решения большинства задач;
— частая корректировка задач даже в режиме функционирования и вытекающая отсюда необходимость изменения и наращивания системы математического обеспечения;
— необходимость автоматизации хода решения задачи для уменьшения вероятности ошибок со стороны оператора, участвующего в управлении вычислительным процессом;

— необходимость организации решения задачи в аварийных ситуациях для обеспечения безусловного решения задачи в заданные сроки.

Анализ задач из класса АСУ позволил выделить ряд типовых процессов обработки информации. К ним относятся: ввод, логический контроль и компоновка информации с разных машинных носителей, сортировка, корректировка информации на магнитных лентах, вывод информации на устройство печати, редактирование, поиск, дублирование информации, логическая обработка нескольких массивов информации.

Естественно, что для каждого из типовых процессов целесообразно иметь процедуру, реализующую соответствующий процесс на ЭВМ. Существует несколько методов реализации таких процедур.

Один из методов заключается в том, что составляется программа типового процесса на одном из алгоритмических языков. В каждом конкретном случае применения этой программы осуществляется корректировка описаний данных и некоторых блоков программы в зависимости от особенностей решаемой задачи.

Вторым методом является разработка универсальной программы для каждого из типовых процессов, при этом настройка соответствующей программы должна осуществляться автоматически по составленным пользователем описаниям и программным блокам.

Третьим методом является разработка генераторов программ. При этом методе на основании описаний пользователя генератор программ автоматически составляет соответствующую рабочую программу, предназначенную для выполнения соответствующего типового процесса над конкретными массивами информации.

В методе использования транслятора с алгоритмического языка трудоемкость привязки программы остается достаточно большой, качество рабочей программы в существенной степени зависит от транслятора, в то же время для разработки такой типовой процедуры требуется сравнительно мало времени при условии, что существует транслятор с выбранного алгоритмического языка.

Метод универсальных программ позволяет сделать привязку очень простой, однако при выполнении программы требуется сравнительно большой объем оперативной памяти.

Метод генераторов программ обладает достоинствами метода универсальных программ, однако требует существенно больших затрат на разработку генератора, и, кроме того, дополнительный этап генерирования требует дополнительных затрат машинного времени.

Рассмотрим реализованные в виде универсальных программ типовые процедуры ввода, контроля и компоновки экономической информации с перфолент и перфокарт, сортировки и корректиров-

ки информации на магнитных лентах, вывода информации на устройство печати и программу управления пакетом задачи.

При разработке рассматриваемых программ учитывался опыт разработки и применения интерпретирующей системы экономического характера (ИСЭ-2) [40]. При разработке всех рассматриваемых программ использовалась система математического обеспечения (СМО) ЭВМ «Минск-32» [23], и как следствие все они ориентированы на обработку информационных массивов, удовлетворяющих требованиям этой системы.

В качестве языка программирования для всех программ использовался язык символического кодирования для ЭВМ «Минск-32».

3.2. ВВОД ЭКОНОМИЧЕСКОЙ ИНФОРМАЦИИ С ПЕРФОЛЕНТЫ

Формальное описание документов, правила их заполнения и перфорации. Основными носителями информации, используемыми при подготовке данных для ввода в ЭВМ, являются первичные документы, составляющие разветвленную информационную систему объекта управления. Каждый документ частично или полностью характеризует определенный объект (явление), представляя собой некоторое множество взаимосвязанных характеристик этого объекта (явления). В документе, кроме количественных характеристик (реквизиты-основания), имеются признаки, определяющие смысловую сущность объекта (реквизиты-признаки) [19]. Документ может иметь сложную иерархическую структуру.

Документ — это некоторая упорядоченная совокупность реквизитов.

Таблица 3.1

	Цех		Мастер		Шифр детали	План
	1		2		3	4
Ц	1	М	1	ДП	А100	1200
Ц		М		ДП	А101	120
Ц		М	2	ДП	В100	500
Ц		М		ДП	А100	1000
Ц	3	М	1	ДП	А101	110

В настоящем разделе описывается структура исходных документов, излагаются правила формального описания документов применительно к программе ввода экономической информации с перфоленты, даются рекомендации по проектированию документов, правила их заполнения и перфорации. Изложение сопровождается иллюстративными примерами (табл. 3.1, табл. 3.2). Пер-

вый из них является сравнительно простым, а второй — достаточно сложным документом. Для простоты изложения обозначим эти документы соответственно Д1 и Д2.

В приведенных примерах реквизитами являются: цех, мастер, шифр изделия, план, вес, норма расхода и др.

Таблица 3.2

Шифр изделия	Вес	Шифр узла	Цех сборки узла	Шифр детали	Количество в узле	Шифр операции	Трудоемкость	Шифр материала	Норма расхода
1	2	3	4	5	6	7	8	9	10
И001	13.5	У015	1	Д135	3	1	13.7	М106	7.6
						3	15.0	М002	13.0
						2	17.5		
				Д015	7	4	15.6	М205	17.0
						5	17.7	М301	134.5
								М001	1.5
И001	13.5	У014	2	Д001	5	4	13.7	М035	1.44
						3	28.5	М145	1.05
								М101	15.7
И002	40.7	У018	3	Д148	9	7	14.7	М148	13.7

Реквизиты в каждом документе, как правило, могут иметь различный уровень подчиненности (в дальнейшем для краткости он будет именоваться уровнем). С помощью младших по уровню реквизитов характеризуются некоторые стороны или свойства старших по уровню реквизитов. Реквизиты могут быть эквивалентными по уровню подчиненности.

Подчиненность реквизитов в документе удобно изображать с помощью графа [4] подчиненности. На рис. 3.1 и рис. 3.2 даны графы подчиненности приведенных ранее примеров документов.

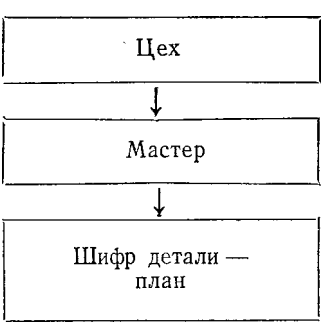


Рис. 3.1. Граф подчиненности документа Д1

Каждой группе эквивалентных реквизитов поставим в соответствие вершину графа подчиненности, стрелки будут изображать подчиненность групп эквивалентных реквизитов.

На одно заполненное в документе значение старшего реквизита может приходиться несколько значений младшего реквизита. Отметим еще одно свойство группы эквивалентных реквизитов: количество заполненных в документе значений эквивалентных реквизитов должно быть одинаковым.

Группы реквизитов, расположенные на одном пути графа подчиненности, будем называть зависимыми; группы реквизитов, расположенные на разных путях — независимыми.

В документе (рис. 3.1) Д1 все группы реквизитов зависимы, это простой документ. В документе (рис. 3.2) Д2 группы реквизитов «шифр операции — трудоемкость» и «шифр материала — норма расхода» зависят от реквизита «шифр детали», но между собой являются независимыми, такие документы будут именоваться сложными.

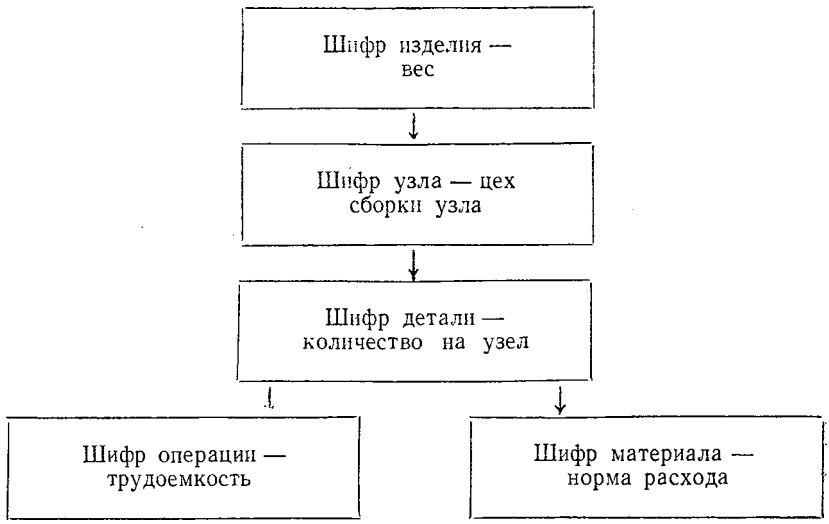


Рис. 3.2. Граф подчиненности документа Д2

Будем называть линейным участком часть графа подчиненности, не содержащую разветвлений. Вершину графа, из которого исходит несколько дуг, будем считать принадлежащей предшествующему линейному участку графа.

Для простых документов весь граф подчиненности представляет собой линейный участок (например, см. рис. 3.1). В документе Д2 (рис. 3.2) линейными участками являются: а) «(шифр изделия — вес) — (шифр узла — цех сборки узла) — (шифр детали — количество на узел)»; б) «шифр операции — трудоемкость»; в) «шифр материала — норма расхода».

Линейные участки, исходящие из последнего узла другого линейного участка, называются линейными участками, подчиненными «верхнему» линейному участку, из которого они исходят.

В правильно спроектированном документе заполнение линейных участков должно вестись с соблюдением следующих правил:

а) эквивалентные по уровню реквизиты либо одновременно заполняются, либо одновременно не заполняются;

б) если заполнен старший реквизит линейного участка, то младшие реквизиты этого линейного участка обязательно должны быть заполнены;

в) если заполнен младший реквизит линейного участка, то старшие реквизиты этого участка не обязательно должны быть заполнены; если некоторый старший реквизит не заполнен, то его значение считается равным последнему заполненному значению этого реквизита в одной из предыдущих строк документа;

г) если заполнен последний реквизит линейного участка, то должен быть заполнен весь набор реквизитов из линейных участков, подчиненных данному, со всеми подчиненными им участками.

Связкой будем называть отрезок линейного участка, реквизиты которого одновременно заполняются и перфорируются, либо не заполняются и не перфорируются.

Связка представляет собой неделимую с точки зрения заполнения и перфорации совокупность реквизитов, последовательно расположенных в исходном документе. Если имеется некоторый линейный участок графа, состоящий из n вершин, то его разбиение на связки может быть, вообще говоря, произведено 2^n способами. Например, в документе Д1 можно выделить четыре варианта разбиения на связки:

а) (цех); (мастер); (шифр детали — план);

б) (цех); (мастер, шифр детали — план);

в) (цех, мастер); (шифр детали — план);

г) (цех, мастер, шифр детали — план).

В документе Д2 линейный участок «шифр изделия — вес, шифр узла — цех сборки узла, шифр детали — количество на узел» может быть разбит на связки аналогично предыдущему примеру. Состоящие из одного узла линейные участки «шифр операции — трудоемкость» и «шифр материала — норма расхода» в любом случае должны быть выделены в самостоятельные связки.

Аналогично понятию подчиненности линейных участков введем понятие подчиненности связок. В пределах одного линейного участка связка, расположенная после другой связки и содержащая младшие по уровню реквизиты, подчинена этой связке и имеет более низкий уровень. В пределах одного линейного участка нумерация уровней производится последовательно по возрастанию (чем ниже уровень, тем больше номер уровня); первая связка самого старшего линейного участка (графа подчиненности) получает номер уровня, равный 1. Старшие связки линейных участков, подчиненных другому участку, получают номер уровня,

на единицу больший, чем номер уровня последней (младшей) связки того участка, которому подчинены данные участки.

Пусть разбиение на связки документа Д1 произведено следующим образом: (цех); (мастер); (шифр детали — план). Тогда связка «цех» имеет уровень 1, связка «мастер» — уровень 2, связка «шифр детали — план» — уровень 3.

Пусть разбиение на связки документа Д2 произведено следующим образом: (шифр изделия — вес, шифр узла — цех сборки узла); (шифр детали — количество на узел); (шифр операции — трудоемкость), (шифр материала — норма расхода). Тогда связка «шифр изделия — вес, шифр узла — цех сборки узла» имеет уровень 1, связка «шифр детали — количество на узел» — уровень 2, связки «шифр операции — трудоемкость» и «шифр материала — норма расхода» — уровень 3.

Понятие связки позволяет преобразовать граф подчиненности, изображающий структуру документа, в другой граф, вершинами которого будут уже не отдельные реквизиты (группы эквивалентных реквизитов), а связки.

Граф связок документа Д1 совпадает с графом подчиненности (рис. 3.1). Если бы в качестве связки были взяты все реквизиты этого простого документа, то граф связок превратился бы в одну вершину.

Граф подчиненности документа Д2 (рис. 3.2) можно преобразовать в граф связок (рис. 3.3).

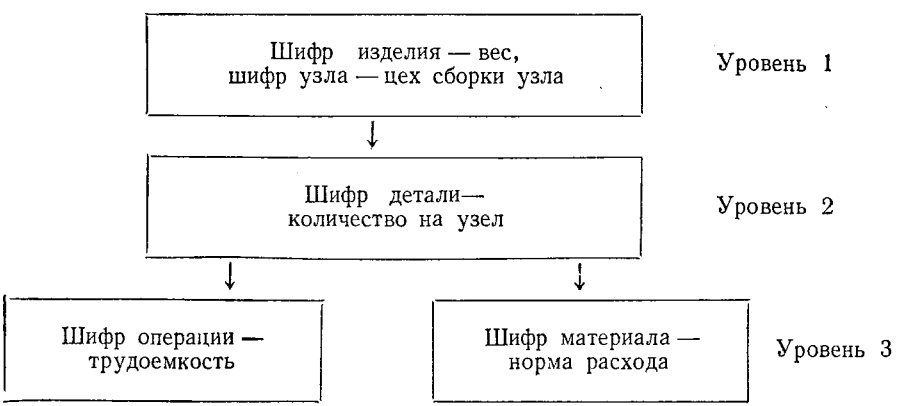


Рис. 3.3. Граф связок документа Д2

Граф связок можно представить с помощью следующего формального описания:

$$i_1C_1,\ i_2C_2,\ \ldots,\ i_nC_n,$$

где i_k — номер уровня связки C_k $(k=1, 2, \ldots, n)$; k — место (номер) связки C_k в последовательности связок документа.

Реквизиты, входящие в связку, как это следует из ее определения, должны располагаться в соседних графах документа. Старшие по уровню связки должны располагаться левее младших, подчиненных им связок.

Последовательность расположения связок в формальном описании документа должна в точности совпадать с порядком расположения соответствующих граф на бланке исходного документа.

Для документа Д1 можно записать:

$$1C_1,\ 2C_2,\ 3C_3,$$

где C_1 — «цех»; C_2 — «мастер»; C_3 — «шифр детали — план».

Для документа Д2:

$$1C_1,\ 2C_2,\ 3C_3,\ 3C_4,$$

где C_1 — «шифр изделия — вес, шифр узла — цех сборки узла»; C_2 — «шифр детали — количество на узел»; C_3 — «шифр операции — трудоемкость»; C_4 — «шифр материала — норма расхода».

Для уменьшения объема заполнения и перфорации информации в документе следует для каждой документо-строки заполнять только те старшие связки, значения реквизитов в которых изменились по сравнению с предшествующей документо-строкой. При этом заполнение старшей связки обязательно влечет за собой необходимость заполнения всех младших и подчиненных связок.

При перфорации значения реквизитов переносятся на последовательный машинный носитель, каким является перфолента, и для восстановления на основе полученной линейной последовательности реквизитов исходной структуры документа необходимо соблюдать следующие правила перфорации:

1) перфорация производится по связкам, т. е. все реквизиты одной связки должны перфорироваться подряд. При этом в начале связки должен перфорироваться ее признак, служащий для опознавания, затем конец признака связки, а после реквизитов — специальный символ — конец связки. Если идентификатор связки отсутствует, то он считается совпадающим с идентификатором предыдущей связки;

2) реквизиты связки перфорируются плотно и разделяются специальным символом — конец реквизита;

3) для связок из линейного участка порядок перфорации реквизитов, входящих в связки, определяется порядком расположения на бланке входного документа и правилом «слева направо, сверху вниз». В частности, это правило для простых документов заключается в следующем: перфорация простых документов должна производиться построчно;

4) после перфорации значений реквизитов младшей связки линейного участка должны перфорироваться все наборы реквизитов тех линейных участков, которые подчинены данному. При этом перфорация реквизитов каждого из подчиненных линейных участков должна вестись «вглубь» до тех пор, пока не будут

отперфорированы реквизиты самого младшего по уровню участка данной цепочки.

Если через $A\{B\}$ обозначить определенное значение связки A и совокупность значений связок B, характеризующих это A, то правило заполнения и перфорации документа Д1 символически можно изобразить следующим образом:

$$\{C_1\{C_2\{C_3\}\}\}.$$

В тех же обозначениях правило заполнения и перфорации документа Д2 символически изображается так:

$$\{C_1\{C_2\{C_3\}\ \{C_4\}\}\}.$$

Проиллюстрируем эти правила для документов Д1 и Д2. При этом служебные символы, необходимые при перфорации, будут изображаться следующим образом: конец реквизита — запятая «,»; конец связки — двоеточие «:»; конец признака связки — открывающая скобка «(». В качестве признаков связок для документа Д1 выбраны начальные буквы наименований реквизитов связок: признак связки 1 — «Ц»; признак связки 2 — «М»; признак связки 3 — «ДП».

После перфорации этот документ будет выглядеть так:

Ц(1:М(1:ДП(А100,1200:А101,120:М(2:ДП(В100,500:
А100,1000:Ц(3:М(1:ДП(А101,110:

Для документа Д2 в качестве признаков связок выберем номера граф документа, где расположены первые реквизиты соответствующих связок: признак связки 1 — «1»; признак связки 2— «5»; признак связки 3 — «7»; признак связки 4 — «9».

После перфорации документ Д2 будет выглядеть следующим образом:

1 (И001, 13.5,У015,1:5(Д135,3:7 (1,13.7:3,15.:
2,17.5:9(М106,7.6:М002,13.0:5(Д015,7:7(4,15.6:
5,17.7:9(М205,17.0:М301,134.5:М001,1.5:
1(И001,13.5,У014,2:5(Д001,5:7(4,13.7:3,28.5:
9(М035,1.44:М145,1.05:М101,15.7:1(И002,40.7,У018,3:
5(Д148,9:7(7,14.7:9(М148,13.7:

На основании изложенного сформулируем порядок подготовки документа с экономической информацией к вводу в ЭВМ:

— определяются уровни подчиненности реквизитов и структура графа подчиненности;

— выделяются связки, определяются их уровни, а также порядок следования и подчинения;

— строится граф подчиненности связок;

— исходя из последовательности связок на бланке документа, а также из порядка их подчинения, отраженного в графе связок, производится формальное описание структуры документа;

— каждой связке присваивается перфорируемый признак (идентификатор) связки и устанавливается порядок заполнения и перфорации документа.

При выделении связок рекомендуется руководствоваться следующими соображениями:

— при уменьшении количества реквизитов, входящих в связку, объем заполнения и перфорации собственно реквизитов уменьшается (за счет опускания старших связок), а объем перфорации служебных символов, необходимых для идентификации связок и реквизитов, возрастает;

— хотя при уменьшении количества реквизитов, входящих в связку, объем заполнения и перфорации, как правило, уменьшается, сложность перфорации и вероятность ошибок в ходе перфорации возрастает за счет необходимости перфорирования служебных символов.

Для упрощения схемы перфорации и выделения связок во входном документе можно вводить идентификаторы связок в шапке документа, выделять разные связки с помощью жирных или цветных линий, вводить в типографский бланк документа служебные символы, подлежащие перфорации. Например, на документе Д1 (табл. 3.1) приведены служебные символы: Ц, М, ДП.

Целесообразно иметь возможность одновременно производить обработку нескольких форм исходных документов, компонуемых в несколько выходных массивов. Эти исходные документы могут находиться на одном физическом машинном носителе. Для того чтобы программа могла различить, какой из исходных документов обрабатывается в данный момент, необходимо указать шифр документа в начале порции данных из этого документа. Для задания этого шифра используется понятие связки нулевого уровня, имеющей стандартный идентификатор ОЛ («операционный лист»). Первым реквизитом операционного листа должен быть шифр документа.

В программе предусмотрена возможность включения шапки документа (т. е. связки самых старших реквизитов) в операционный лист, при этом она помещается после шифра документа.

Связка типа «операционный лист» не является обязательной. Она должна быть, если предусматривается одновременная обработка нескольких документов различной структуры или если реквизиты из шапки документа вынесены в операционный лист.

Рассмотрим универсальную программу ввода экономической информации с перфоленты * на ЭВМ «Минск-32».

Эта программа состоит из двух автономных модулей. Первый из них — программа обработки описаний (ОБРОД) — предназначен для обработки описаний пользователя и приведения их к виду, удобному для работы основной программы. Второй — соб-

---

* В разработке программы принимали участие В. Н. Агафонов, В. Ю. Боболович.

ственно программа ввода экономической информации с перфоленты, (ВВОДЛ) который, пользуясь полученными с помощью первой программы машинными описаниями, осуществляет ввод, контроль и компоновку информации с перфоленты.

Программа обработки описаний (ОБРОД). Данная программа является вспомогательной для основной программы ввода экономической информации с перфоленты и предназначена для обработки (трансляции) описаний, составленных пользователем, в вид, пригодный для дальнейшего их использования основной программой. Блок-схема программы приведена на рис. 3.4. Программа вводит описания, подготовленные на перфоленте или перфокартах, и перерабатывает их в машинный вид, требуемый основной программой.

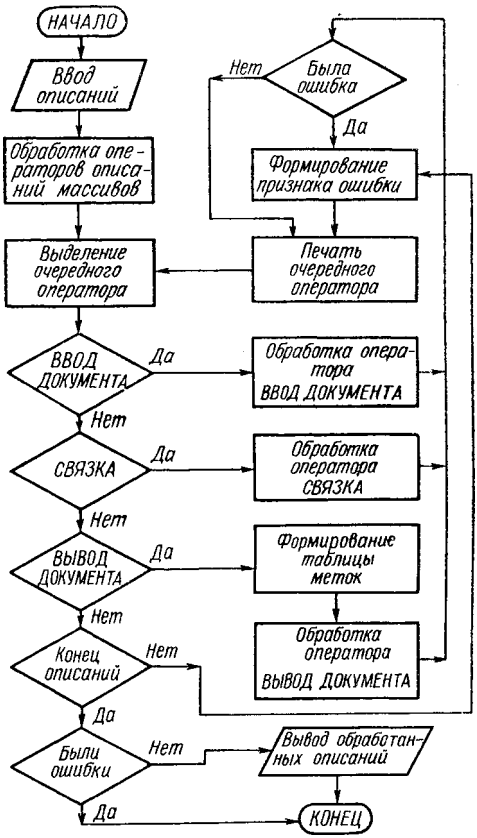


Рис. 3.4. Блок-схема программы ОБРОД

Структура описаний. Описания в общем случае состоят из следующих составных частей: описаний массивов, описаний входных документов и разделителей, описаний выходных записей.

Отметим, что после описания входного документа должны следовать описания выходных записей, компонуемых из данных соответствующего входного документа.

Не накладывается ограничений на количество входных документов, а также на количество записей, компонуемых на основе любого входного документа.

Описания выходных записей могут или отсутствовать полностью, или отсутствовать только после некоторых из описаний входных документов. В последнем случае во время компоновки выходные записи компонуются только из документов, после описаний которых следуют описания записей.

Все описания состоят из отдельных операторов.

Описание массивов организуется с помощью операторов ВВОД МАССИВА, РАБОЧИЙ МАССИВ и ВЫВОД МАССИВОВ.

<оператор ВВОД МАССИВА*>::=ВВОД␣МАССИВА:<описание входного массива>;

<описание входного массива>::=<имя массива> (<сведения о входном массиве>)

<сведения о входном массиве>::=<носитель>|<сведение о МЛ>

<носитель>::=МЛ|ПЛ

<сведение о МЛ>::=МЛ—<имя МЛ>

<оператор РАБОЧИЙ МАССИВ>::=РАБОЧИЙ␣МАССИВ:<описание рабочего массива>;

<описание рабочего массива>::=<имя массива>|<имя массива> (сведения о массиве)

<сведения о массиве>::=<сведения о МЛ>|<сведения о МЛ>, <срок годности>

<оператор ВЫВОД МАССИВОВ>::=ВЫВОД␣МАССИВОВ:<описания выходных массивов>;

<описания выходных массивов>::=<описание выходного массива> | <описание выходного массива>, <описание выходного массива > | ∠описание выходного массива>, <описание выходного массива>, <описание выходного массива>

<описание выходного массива>::=<имя массива>|<имя массива> (сведения о выходном массиве)

<сведения о выходном массиве>::=<сведения о массиве>|<сведения о записи>|<сведения о записи>, <сведения о массиве>

<сведения о записи>::=<формат записи>|<формат записи><длина зоны>

<формат записи>::=Ф|П

∠длина зоны>::=<целое без знака>

Оператор ВВОД МАССИВА служит для описания входного массива. Если ввод исходной информации осуществляется с МЛ, то наличие оператора ВЫВОД МАССИВОВ необходимо. В этом случае при отсутствии карт заказа программа ВВОДЛ (см. ниже) будет работать только в режиме компоновки. Оператор ВЫВОД МАССИВОВ служит для описания выходных массивов скомпонованных записей. Этот оператор может отсутствовать, тогда компоновка производиться не будет. Оператор РАБОЧИЙ МАССИВ служит для описания массива, выводимого на МЛ пос-

---

* Здесь и далее используется символика металингвистических формул Бэкуса [1]. Символ «␣» обозначает пробел. Если он не указан, то соответствующие данные должны быть записаны без пробелов.

ле контроля. Этот оператор может отсутствовать, в этом случае вывод на МЛ после контроля производиться не будет.

В качестве имени массива и имени МЛ могут служить любые 5 символов. Имена компонуемых массивов должны быть различными. Если сведения о массиве отсутствуют в одном из операторов, программа ВВОДЛ не будет осуществлять проверку, та ли магнитная лента установлена.

В качестве формата записи может служить один из символов «П» или «Ф». Если употреблен символ «П», то вывод скомпонованных записей на магнитную ленту будет производиться записями переменной длины; если «Ф»,— то записями фиксированной длины. Если формат записи отсутствует (в этом случае должна отсутствовать и длина зоны), то вывод происходит записями фиксированной длины.

Кроме формата записей, пользователь может указать длину информационной части зоны. Она задается десятичным числом ячеек. Если длина не указана, вывод будет производиться стандартными зонами длиной 320 ячеек (1920 информационных символов). На длины зон, указываемые пользователями, не накладывается никаких ограничений.

В операторе ВЫВОД МАССИВОВ может быть описано не более трех массивов.

Описание разделителей задается с помощью оператора СИМВО.

<оператор СИМВО>::=СИМВО:<конец зоны><конец связки><конец реквизита><конец признака связки><зачеркивание>

Каждой из компонент этого оператора соответствует один символ. В результате после двоеточия должно быть пять символов, используемых как служебные при организации исходных массивов экономической информации на перфоленте.

Если какой-либо из служебных символов является стандартным (на который настроена программа), то вместо этого символа в операторе перфорируется код пробела. Это дает возможность заменить часть стандартных символов, так как написание в операторе стандартных служебных символов недопустимо.

Кроме того, в роли служебного запрещается использование символа пробела и совпадение двух служебных символов недопустимо.

Символ «Конец зоны» на перфоленте может не перфорироваться, но он все равно вставляется основной программой.

Оператор СИМВО может отсутствовать. В этом случае используются стандартные служебные символы, другими словами, в программе предусмотрен стандартный вид этого оператора:

СИМВО:↑;,)!

Предусмотрены еще две возможности замены разделителей с помощью карт заказа или заданием при обращении во внутрен-

нем режиме. Символ «)» можно использовать только в качестве разделителя «Конец признака связки».

Описание входного документа организуется с помощью операторов ВВОД ДОКУМЕНТА и СВЯЗКА, структура которых приведена ниже.

<оператор ВВОД ДОКУМЕНТА>::=ВВОД ␣ДОКУМЕНТА␣
<шифр документа> : <связки>;
<шифр документа>::=<5 любых символов>
<связки>::=<связка>|<связка>, <связки>
<связка>::=ОЛ|<уровень связки><имя связки>|
<уровень связки><имя связки> (<границы связки>)
<границы связки>::=<МИН>, <МАКС>|<МИН>
<оператор СВЯЗКА>::=СВЯЗКА␣<информация о связке>:
<В-реквизиты>;
<информация о связке>::=ОЛ|<имя связки> (<признак связки>)
<В-реквизиты> : :=<В-реквизит>| <В-реквизит>, <В-реквизиты>
<В-реквизит> : :=<имя реквизита> (<тип В-реквизита>, <границы реквизита>) | <имя реквизита> (<тип В-реквизита>)
<границы реквизита>::=<НГ>, <ВГ>|<ВГ>
<тип В-реквизита>::=Т|В|Ц

Типы: Т — текстовый; В — число со знаком или без, с десятичной точкой или без; Ц — число целое без десятичной точки.

В описании каждого входного документа оператор ВВОД ДОКУМЕНТА встречается только один раз, оператор СВЯЗКА — столько раз, сколько наименований связок, включая и ОЛ, встречается в операторе ВВОД ДОКУМЕНТА. Порядок следования операторов СВЯЗКА должен соответствовать порядку следования наименований связок в операторе ВВОД ДОКУМЕНТА.

В качестве наименований связок и реквизитов можно использовать любое количество символов. Программа использует из наименований не более пяти первых символов. Наименования всех связок и реквизитов должны быть различны и начинаться с буквы.

Под верхней и нижней границами связок (соответственно МИН, МАКС) понимаются максимальное и минимальное допустимые количества однородных связок, которые могут следовать за общей для них старшей связкой. Границы могут отсутствовать. Если МИН=0, то соответствующая связка называется нулевой, и она в некоторых исходных документах может отсутствовать со всеми ей подчиненными связками.

Каждый числовой реквизит (типа В или Ц) по величине должен быть не менее нижней (НГ) и не более верхней (ВГ) границ, указанных в описании для этого реквизита. Границы могут указываться со знаком. НГ должна быть не больше, чем ВГ.

Если границы не указаны, то контроль на диапазон изменения соответствующего реквизита производиться не будет.

Границы текстового реквизита (НГ и ВГ) означают максимально и минимально допустимые количества символов в реквизите.

Описание выходной записи производится с помощью оператора ВЫВОД ДОКУМЕНТА.

<оператор ВЫВОД ДОКУМЕНТА>::=ВЫВОД␣ДОКУМЕНТА␣<информация о записи>:<реквизиты>;
<информация о записи>::=(<длина записи>)␣В␣МАССИВ␣<имя массива>|В␣МАССИВ␣<имя массива>
<реквизиты>::=<реквизит>|<реквизит>, <реквизиты>
<реквизит>::=<имя реквизита> (<параметры реквизита>)|<литерал> (<номер ячейки>, <номер символа>)
<параметры реквизита>::=<Т-тип>, <номер ячейки>, <номер символа>,
<количество символов>|<В-тип>, <номер ячейки>, <номер разряда>, <количество разрядов>|<Ц-тип>, <номер ячейки>, <номер разряда>, <количество тетрад>|<П-тип>, <номер ячейки>
<Т-тип>::=Т
<В-тип>::=ВБ|ВЗ
<Ц-тип>::=ДЦ|ДЦ<масштаб>
<П-тип>::=ПЛ|ВФ|ДФ
<масштаб>::=<цифра>

Если длина скомпонованной записи не приводится, то программа приписывает записи ее реальную длину. Если длина указана, то она должна быть не менее реальной длины. Если больше, то свободные ячейки в конце записи заполняются нулями. В качестве литерала могут служить не более пяти символов, которые без изменений переносятся в запись.

Предусмотрены следующие типы реквизитов, компонуемых в запись:

Т — текстовый. Информация записывается в виде текста. Исходный реквизит текстовый или числовой без изменений переносится в отведенный для него участок записи. Если длина исходного реквизита больше длины участка, то для текстового реквизита отбрасываются пробелы, стоящие спереди, а затем, если этого недостаточно, переносится столько символов, сколько поместится, а последние отбрасываются; для числового исходного реквизита отбрасываются символы в такой последовательности: нули в начале и знак плюс, цифры после десятичной точки, начиная с конца, и десятичная точка, если она есть, и, наконец, первые значащие цифры — до тех пор, пока реквизит не поместится в участок. Если исходный реквизит не занимает всего участка, то текстовый реквизит дополняется в конце пробелами, а числовой в начале — нулями.

ВБ — восьмеричный целый без знака. Модуль исходного реквизита переводится в двоичную систему и записывается в отведенный участок. Если переведенный реквизит не помещается, то отбрасывается спереди необходимое число двоичных разрядов. Переведенный реквизит может занимать всю ячейку или часть ее.

ВЗ — восьмеричный целый со знаком. Аналогичен типу ВБ, только первый разряд из выделенных под этот реквизит используется как знаковый. Он равен нулю, если исходный реквизит был с плюсом, и 1— в противном случае.

ВФ — восьмеричный с фиксированной запятой. Занимает целую ячейку. Исходный реквизит по модулю не должен превосходить 1. В описании пользователя для этого реквизита должны быть обязательно указаны границы, которые по модулю меньше 1.

ДЦ — десятичный целый. Может содержать не более 99 тетрад и занимать всю ячейку, часть ее или несколько ячеек. В последнем случае последняя ячейка, отведенная под реквизит, может быть занята не полностью. Если исходный реквизит не помещается в отведенный участок, то отбрасывается необходимое количество первых цифр. Нулевой разряд ячеек, занимаемых реквизитом, остается не занятым. Если реквизит начинается с нулевого разряда, то этот разряд используется только как знаковый, а первая тетрада будет начинаться с первого разряда. Если реквизит начинается с любого другого разряда, то знак исходного реквизита опускается. Если реквизит занимает несколько ячеек, то начинаться в первой ячейке он должен с такого разряда, чтобы последняя тетрада в этой ячейке занимала разряды 33÷36.

Предусмотрена возможность компоновки реквизитов типа ДЦ <масштаб>. В этом случае сразу после символов ДЦ записывается величина масштаба, которая занимает не более одного символа. Величина масштаба может принимать значение от 1 до 9.

Если указана величина масштаба k, то предварительно реквизит умножается на 10^k и в дальнейшей обработке используется только целая часть полученного числа.

ДФ — десятичный с фиксированной запятой. Аналогичен типу ВФ, только в двоично-десятичной системе счисления.

ПЛ — двоичный с плавающей запятой. Занимает целую ячейку.

Предусмотрены следующие виды редактирования исходных реквизитов:

Т → Т

Ц → Т, ВБ, ВЗ, ДЦ, ПЛ

В → Т, ВФ, ДФ, ПЛ и ДЦ с масштабом.

В описаниях возможно применение оператора «комментарии». Этот оператор должен начинаться кодом «[», он в обработке не используется, а выводится только на печать.

Описания пользователя составляются в произвольной форме и перфорируются на перфоленту или перфокарты. Оформление массива при перфорации на перфоленту или перфокарту должно удовлетворять требованиям СМО «Минск-32».

При перфорации на перфоленту все операторы перфорируются последовательно один за другим. В конце каждого оператора перфорируется символ «;». В конце массива описаний перфорируется символ «?».

Если при перфорации на перфоленту после символа «;» был отперфорирован символ «!», то последний оператор считается зачеркнутым. Если после любого другого символа отперфорирован один или несколько символов «!», то считаются зачеркнутыми столько же последних информационных символов последнего оператора.

При перфорации на перфокартах описания записываются на бланках ЯСК. Каждый оператор должен начинаться с 12-й позиции новой строки. Если оператор не помещается на одной строке, то он может быть продолжен на одну или несколько других строк. Продолжение на новой строке должно начинаться с 13-й позиции; в этом случае 12-я позиция не заполняется, а при перфорации пропускается. Иначе говоря, 12-я колонка перфокарты с продолжением должна содержать код пустой колонки. Необязательно, чтобы перед переносом строка была полностью заполнена. Символ «зачеркивание» на картах не применяется. Описание, составленное пользователем, перерабатывается (транслируется) программой обработки описаний (ОБРОД). В процессе обработки производится синтаксический и семантический контроль описаний, который заключается в проверке описанных ниже условий.

Если входной массив включает документы различной структуры (в описании встречается несколько операторов ВВОД ДОКУМЕНТА), то первой связкой каждого документа должен быть операционный лист (связка ОЛ).

Перед именами связок в операторе ВВОД ДОКУМЕНТА должен быть указан уровень связок (исключением является связка ОЛ).

Вслед за связкой уровня k ($k \geqslant 1$) в операторе ВВОД ДОКУМЕНТА может следовать связка уровня l, при этом обязательно $1 < l \leqslant k+1$.

Различные связки (реквизиты) должны иметь различные имена. Связки описываются (оператор СВЯЗКА) в той последовательности, в которой они следуют в соответствующем операторе ВВОД ДОКУМЕНТА. Если носителем исходной информации указана МЛ, то оператор РАБОЧИЙ МАССИВ отсутствует. Шифры документов в различных по структуре входных документах должны быть различными.

В описании компонуемых реквизитов и литералов оператора ВЫВОД ДОКУМЕНТА количество параметров, а также их значения должны соответствовать типу реквизита (номер символа

5, номер разряда ⩽37 и т. д.) Компонуемые массивы, имя которых указывается в операторе ВЫВОД ДОКУМЕНТА, а также имена реквизитов в этом операторе должны быть описаны в операторах ВЫВОД МАССИВОВ и СВЯЗКА. Разные связки должны иметь разные признаки начала связки. Тип компонуемого реквизита должен соответствовать типу исходного реквизита, например, если тип исходного реквизита Т, то тип компонуемого реквизита не может быть ДЦ, ПЛ и т. д. Поля, отведенные под компонуемые реквизиты в выходных записях, не должны пересекаться. В одну выходную запись не могут компоноваться реквизиты из различных независимых связок. Все параметры в операторах должны быть корректно представлены, например, такие параметры, как МИН, МАКС и уровень связки (оператор ВВОД ДОКУМЕНТА) должны быть представлены кодами цифр, параметр «тип В-реквизита» (оператор ВВОД ДОКУМЕНТА) может принимать одно из значений Т, В, Ц. В операторе ВВОД ДОКУМЕНТА МИН⩽МАКС, а в операторе СВЯЗКА НГ⩽ВГ. В операторе ВЫВОД МАССИВОВ может быть описано не более трех выходных массивов.

Переработанные (машинные) описания выводятся на перфоленту или магнитную ленту. Вывод на магнитную ленту осуществляется одной зоной. Кроме того, на печать выдаются исходные описания с указанием всех ошибок, если последние были обнаружены. Если описания вводились с перфокарт, то предварительно распечатывается содержимое каждой перфокарты. После распечатки исходных описаний на печать выдаются:

длина машинного описания;

длина поля, в котором будут формироваться скомпонованные записи;

длины полей обмена (или длины зон, увеличенные на 3 ячейки) для вывода скомпонованных записей.

Обращение к программе обработки описаний осуществляется через программу КООРДИНАТОР:

ВЫ-АААААА;ПМ◊ *АААААА*ОБРОДООЗООЛСО22О1◊

Пример. Имеется входной документ, форма которого следующая:

Изделие		Деталь	
шифр	количество	шифр	количество

Реквизиты в данном документе удовлетворяют следующим условиям:

Полное наименование реквизита	Имя реквизита	Тип	Нижняя граница (НГ)	Верхняя граница(ВГ)
Шифр изделия	ШИЗД	Т	1	7
Количество изделий	КОЛИЗД	Ц	1	100
Шифр детали	ШДЕТ	Т	1	7
Количество деталей	КОЛДЕТ	Ц	1	100

Связки удовлетворяют следующим условиям:

Имя связки	Уровень связки	Признак связки	Границы	
			МИН	МАКС
ИЗДЕЛИЕ	1	И	—	—
ДЕТАЛЬ	2	Д	100	200

Необходимо скомпоновать записи следующей структуры:

ШИЗД
КОЛИЗД
ШДЕТ
КОЛДЕТ

Допустим, что реквизиты компонуемой записи удовлетворяют следующим условиям:

Имя реквизита	Тип реквизита	Номер ячейки	Номер символа (разряда)	Количество символов (тетрад)
ШИЗД	Т	1	0	5
КОЛИЗД	ДЦ	2	0	9
ШДЕТ	Т	3	0	5
КОЛДЕТ	ДЦ	4	0	9

Имена массивов:
— исходного на перфоленте — МАСС1;
— компонуемых записей — КОМПД.

Составляется следующее описание:

ВВОД␣МАССИВА : МАСС1 (ПЛ);
ВЫВОД␣МАССИВОВ : КОМПД;
ВВОД␣ДОКУМЕНТА␣ДОК01: 1ИЗДЕЛИЕ 2ДЕТАЛЬ(100,200);
СВЯЗКА␣ИЗДЕЛИЕ (И) : ШИЗД(Т,1,7), КОЛИЗД(Ц,1,100);
СВЯЗКА␣ДЕТАЛЬ(Д) : ШДЕТ(Т,1,7),КОЛДЕТ(Ц,1,100);
ВЫВОД␣ДОКУМЕНТА(4)␣В␣МАССИВ МАСС1 : ШДЕТ(Т,3,0,5);
ШИЗД(Т,1,0,5),КОЛИЗД(ДЦ,2,0,9)КОЛДЕТ(ДЦ,4,0,9); ↑

Данные описания перфорируются на перфоленте и обрабатываются программой обработки описаний. Обработанное описание выводится на магнитную ленту или перфоленту.

Программа ввода с перфоленты (ВВОДЛ). Данная программа является основной и предназначена для реализации ввода, контроля и компоновки экономической информации с исходных (первичных) документов. В качестве промежуточного носителя используется перфолента.

В программе предусмотрены следующие режимы работы:

— контроль экономической информации, введенной с перфоленты или с магнитной ленты, без компоновки;
— контроль экономической информации, введенной с перфоленты или с магнитной ленты, с компоновкой (при наличии компоновки выходной массив записывается на магнитную ленту);
— контроль экономической информации, введенной с перфоленты, с записью на магнитную ленту без компоновки;
— контроль экономической информации, введенной с перфоленты, с записью на магнитную ленту и с компоновкой;
— компоновка информации, введенной с перфоленты или магнитной ленты, без контроля;
— запись экономической информации, введенной с перфоленты, на магнитную ленту без контроля и компоновки.

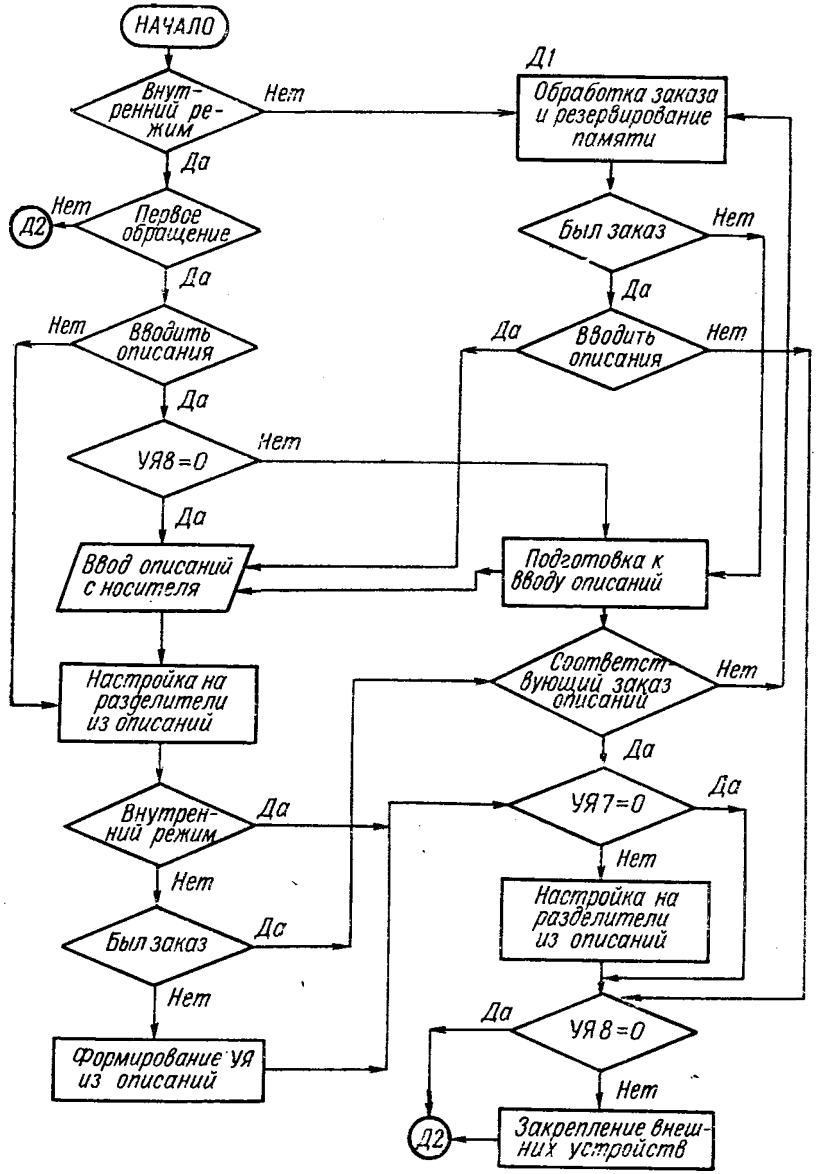
Блок-схемы программы приведены на рис. 3.5, 3.6, 3.7. На них используются следующие обозначения:

УЯ — управляющая ячейка;
УЯi — i-й разряд управляющей ячейки;
ОЛ — операционный лист;
ПОРС — порядок следования;
КОЛКД — количество компонуемых записей из данного входного документа;
КОЛРЕ — количество реквизитов в компонуемой записи;
ОВХД — описание входного документа;
ОКОР — описание компонуемого реквизита;
УОСВ — уровень обрабатываемой связки;
УПРСВ — уровень предыдущей связки.

Программа использует:
— описания, подготовленные с помощью вспомогательной программы, находящиеся на перфоленте или магнитной ленте;
— массивы исходных документов с экономической информацией, подготовленные описанным ниже способом на перфоленте;
— информацию о режиме работы программы, передаваемую через карты заказа (при обращении к ВВОДЛ в режиме внешней программы) или указанную в головной программе (при обращении к ВВОДЛ во внутреннем режиме).

Документы, предназначенные для обработки основной программой, перфорируются на перфоленте. При этом предполагается, что первичные документы подготовлены к вводу по правилам, описанным ранее.

Рассмотрим вариант, когда перфорация исходной информации производится в коде ГОСТ 10859—64. В конце каждой связки перфорируется разделитель «конец связки». После каждого реквизита, кроме последнего в связке, перфорируется разделитель «конец реквизита».



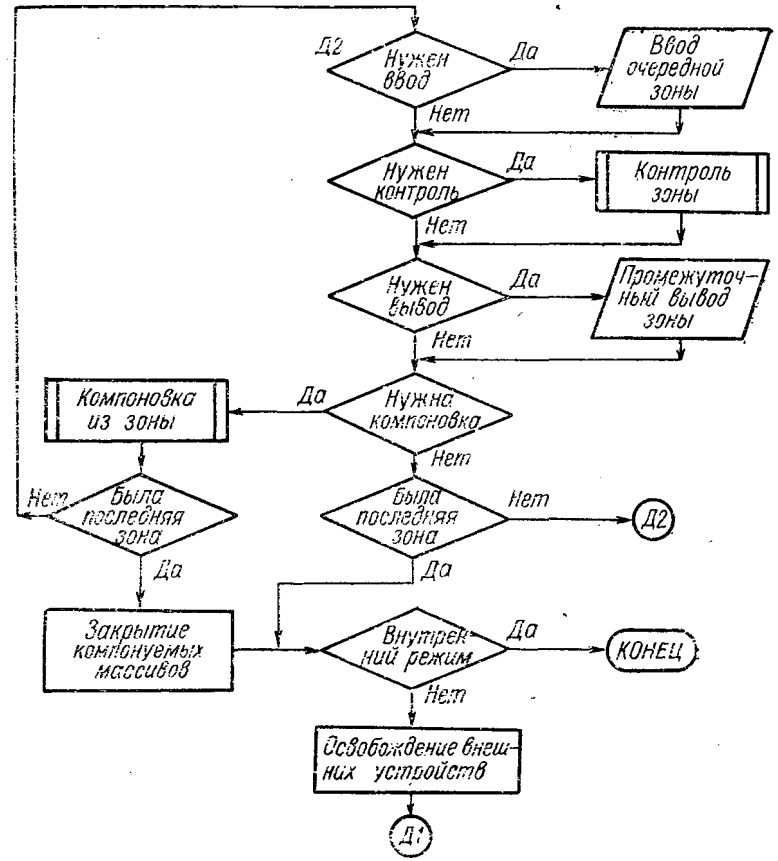


Рис. 3.5. Блок-схема программы ВВОДЛ

Кроме того, используется служебный символ «зачеркивание». Если этот символ отперфорирован после разделителя «конец связки», то считается зачеркнутой вся связка. Если этот символ (один или несколько) отперфорирован после любого информационного символа или разделителя «конец реквизита», то считаются зачеркнутыми столько последних отперфорированных реквизи-

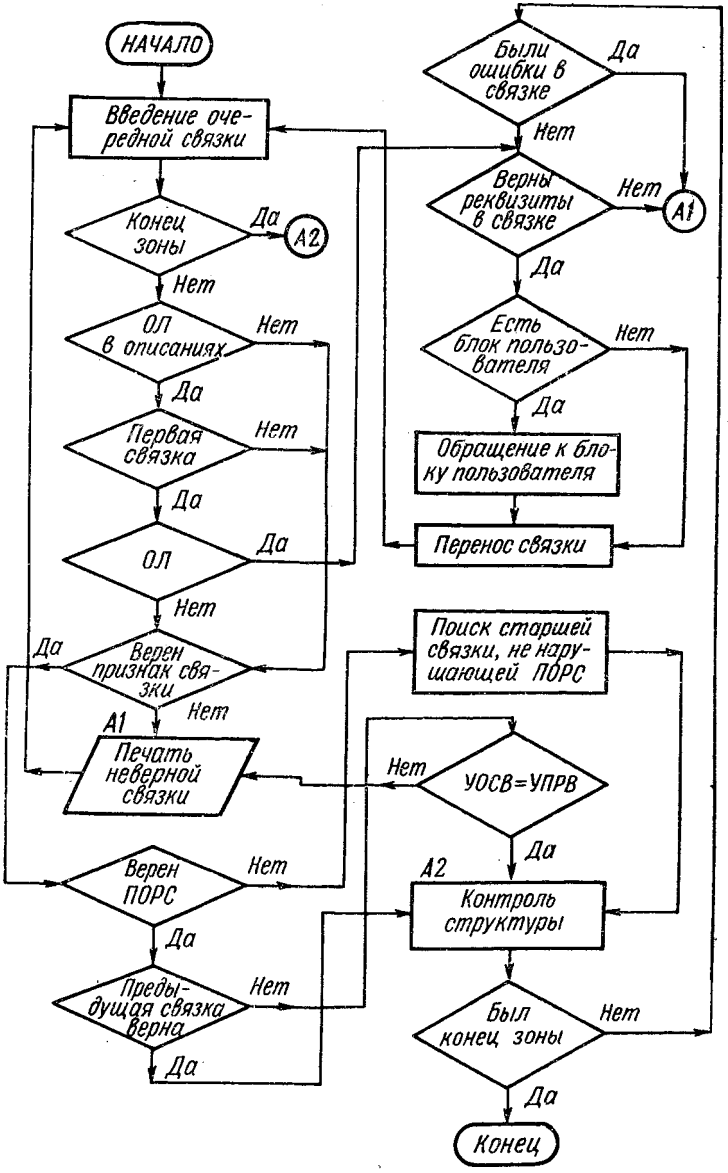


Рис. 3.6. Блок-схема блока «Контроль зоны»

тов в связке, сколько отперфорировано символов «зачеркивание».

В начале связки должен быть ее признак, после которого перфорируется разделитель «конец признака». Признак может состоять из любого количества символов. Если подряд перфорируется последовательность связок, имеющих одинаковый приз-

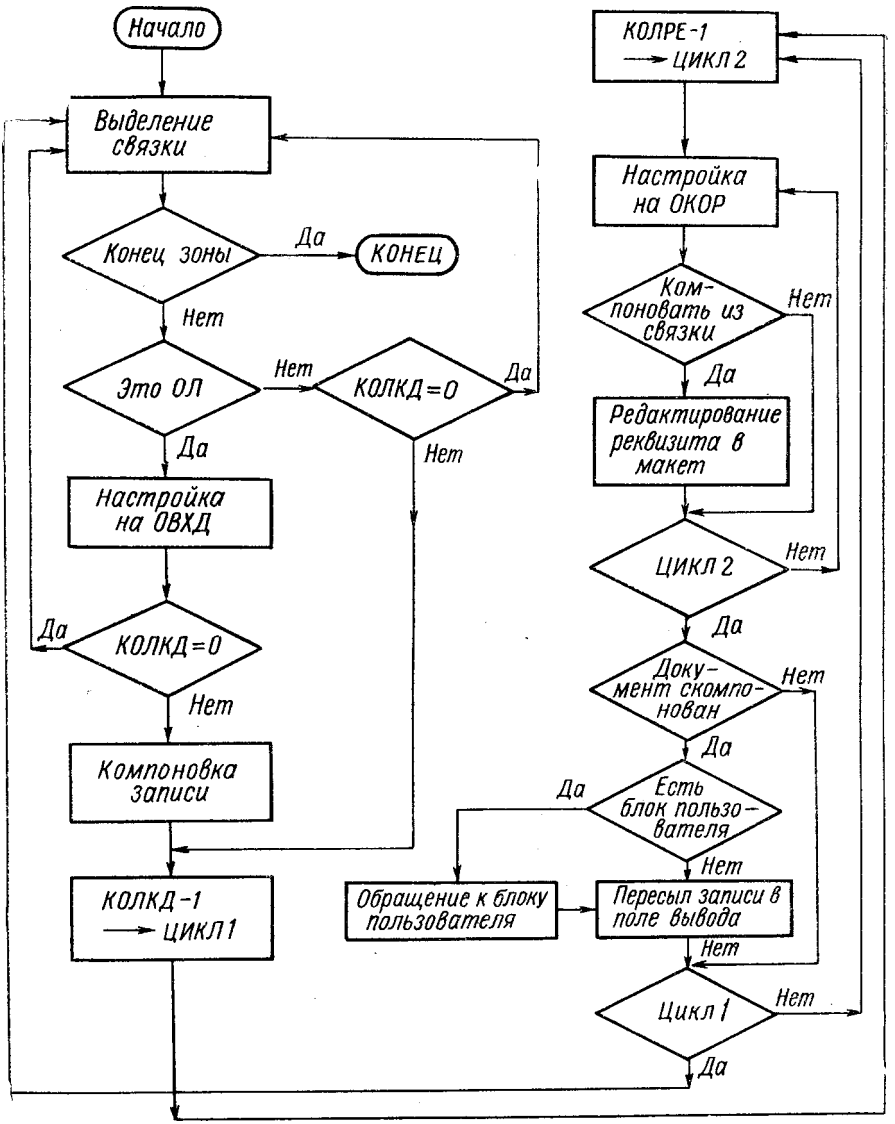


Рис. 3.7. Блок-схема блока «Компоновка из зоны»

нак, то признак достаточно отперфорировать только при первой из них.

Программа настроена на стандартные служебные символы. В качестве разделителя «конец связки» используется символ «;», разделителя «конец реквизита» — символ «,», разделителя «конец признака связки» — символ «)», разделителя «зачеркивание» — символ «!». Однако в программе предусмотрена возможность замены вышеперечисленных служебных символов полностью или частично символами, удобными для пользователя. Эти новые служебные символы пользователь должен указать или в описаниях, или при обращении во внутреннем режиме — путем записи адресов ячеек с новыми разделителями, или при обращении в режиме отдельной программы — с помощью карт заказа. Служебные символы не должны использоваться как информационные.

В программе предусмотрена возможность обработки многокатушечных массивов. Оформление как одно-, так и многокатушечных массивов на перфоленте должно удовлетворять требованиям СМО ЭВМ «Минск-32».

Программа ввода с перфоленты может работать как во внутреннем, так и во внешнем режиме. Вся работа программы организуется на основании управляющей ячейки, которая задается пользователем в режиме внутренней программы, или формируется программно на основании карт заказа или описаний документов в режиме внешней программы. В управляющей ячейке собраны сведения о режиме работы программы.

В режиме внешней программы организуется проверка, не требует ли пользователь с помощью карт заказа действий, не предусмотренных в описаниях. Если подобное обнаружено, то дальнейшая работа программы прекращается и организуется переход к вводу очередного заказа.

На основании разделителей, указанных в описаниях, производится настройка программы. Если разделители дополнительно задаются пользователем, то разделители из описаний игнорируются.

После ввода описаний в режиме отдельной программы или при модифицированном обращении в режиме внутренней программы производится динамическое распределение памяти и закрепление внешних устройств.

Дальнейшая работа строится следующим образом. После ввода каждой зоны исходной информации на основании данных управляющей ячейки выполняются некоторые из следующих процедур: контроль зоны, промежуточный вывод на магнитную ленту и компоновка. Если введена последняя зона, происходит закрытие всех выходных массивов. Если ввод исходной информации производится с перфоленты, то после ввода зона подвергается предварительной обработке. При этом выполняются следующие функции:

— выбрасываются из связки зачеркнутые реквизиты;
— выбрасываются все зачеркнутые связки;
— если ошибочно отперфорировано несколько символов-разделителей, то все символы, кроме первого, выбрасываются.

Контроль зоны организуется следующим образом. В каждой связке проверяется, присутствует ли признак. Если признака нет, то связке приписывается признак предыдущей. Если связка имеет признак, то выделяется не более пяти первых символов признака и начинается поиск в описании соответствующего входного документа описания связки, у которой такой же признак. Если такого описания не обнаружено, то исключаются из обработки и выводятся на печать все связки до тех пор, пока не встретится связка первого уровня или операционный лист. Затем на основании таблицы меток предыдущей связки организуется проверка, не нарушает ли обрабатываемая связка порядка следования. Если порядок следования нарушен, то из обработки исключаются и выводятся на печать данная связка и все следующие за ней, пока не встретится связка, уровень которой старше или равен уровню этой связки и которая не нарушает порядка следования.

Если уровень обрабатываемой связки старше уровня предыдущей, то организуется проверка структуры. Вся неверная ветвь выводится на печать и исключается из дальнейшей обработки.

Далее контролируется соответствие количества реквизитов в связке количеству, указанному в описании связки. Реквизиты связки контролируются на корректность представления (первым символом реквизита типа Ц может быть знак + или —, все остальные символы должны быть цифрами; входной реквизит типа В в отличие от типа Ц может содержать код десятичной точки), на диапазон изменения (НГ$\leqslant$<реквизит>$\leqslant$ВГ).

В случае обнаружения ошибки в связке эта связка и все ей подчиненные исключаются из дальнейшей обработки.

Если в связке не было обнаружено ошибок, то во внутреннем режиме организуется обращение к блоку пользователя (если такой есть). В блоке пользователя может быть проведен дополнительный контроль вводимой информации.

Блок компоновки выделяет из каждой связки нужные реквизиты, осуществляет их редактирование и занесение в макеты компонуемых записей. После обработки каждой связки проверяется, полностью ли скомпонованы выходные записи.

Полностью скомпонованные записи выводятся на магнитную ленту. Перед выводом во внутреннем режиме организуется обращение к блоку пользователя, если такой есть.

Результатом работы основной программы является образование не более трех массивов скомпонованных записей, записанных на различных магнитных лентах по правилам организации обмена с внешними устройствами для ЭВМ «Минск-32». Различные по структуре скомпонованные записи, каждая из которых имеет по-

стоянную для своего типа длину, могут по желанию пользователя выводиться в один или разные массивы записями переменной или фиксированной длины. Если в один массив компонуются различные по структуре записи, то их длины должны быть равны (при выводе записями фиксированной длины) и могут быть различными (при выводе записями переменной длины). Вывод производится зонами фиксированной длины. Длина зоны для каждого массива задается пользователем.

Кроме того, в программе предусмотрен вывод на магнитную ленту исходной информации, введенной с перфоленты. В этом случае вывод производится зонами переменной длины, длина каждой из которых определяется длиной соответствующей ей бобины перфоленты. Если перед выводом осуществлялся контроль, то в каждой зоне будет оставаться только верная информация из соответствующей бобины перфоленты.

В процессе контроля на печать выдаются все операционные листы, а также ошибочные связки, обнаруженные в информации. Для облегчения поиска неверной связки в исходной информации перед печатью каждой неверной связки выдается на печать цепочка старших связок, которым неверная связка подчинена.

После каждой неверной связки выводятся на печать все подчиненные ей связки, которые также исключаются из дальнейшей обработки. Для каждой связки печатаются ее имя, уровень и, если надо, признак ошибки. Признак ошибки (символ «*») означает, что данная связка выброшена из массива. Кроме того, для неверной связки печатается вид ошибки.

Сообщение на АЦПУ имеет вид:

ПРИЗНАК ОШИБКИ	ПВ	Имя связки	УРОВ	Связка

В графе ПВ печатается признак неверной связки «*»; в графе УРОВ — уровень связки; в графе ПРИЗНАК ОШИБКИ — тип ошибки.

Обращение к программе в режиме отдельной программы. При запуске программы с помощью КООРДИНАТОРа (режим отдельной программы) возможно использование карт заказа в том случае, когда необязательно выполнять полный объем работы, заданный описаниями. При выполнении полного объема работ можно работать без карт заказа.

Заказ на работу программы (вид его указан ниже) записывается на бланках ССК и затем перфорируется на перфокарты.

Заказ на выполнение имеет структуру, приведенную в табл. 3.3.

Оператор ИМНОД служит для задания имени ОД, а также носителя, на котором находится ОД. Он может отсутствовать в заказе только при перезаписи информации с ПЛ на МЛ.

*Таблица 3.3**

Этикетка	КОП	Адреса и замечания
ИМНОД	<имя массива ОД>	{ПЛ / МЛ / МЛ—<имя МЛ>}
ВЫП	КОНТР	{З / <пусто>}
ВЫП	КОМПО	{1 / 0} {1 / 0} {1 / 0}
ВЫП	ПРОМЫ	
РЗД	{<разделители> / <пусто>}	
НЭИ	{МЛ / ПЛ}	{<пусто> / З}
ОПИСВ	<имя массива>	{<имя МЛ> / <пусто>}
ОПИСЫ	<имя массива>	{<имя МЛ><срок годности> / <имя МЛ><срок годности>}

Оператор ВЫП служит для задания режимов работы программы ВВОДЛ. Присутствие хотя бы одного ВЫП в заказе обязательно. ВЫП КОНТР требует выполнения процедуры контроля. В этом случае в 22-й позиции может быть указан символ «З», что означает запрещение контроля структуры. ВЫП КОМПО требует выполнения процедуры компоновки. В 22—24-й позициях этого оператора могут быть указаны символы 1 или 0, 1 указывает на разрешение компоновать соответствующий массив, описанный в ОД (соответствие по порядку следования). Если эти позиции не заняты, то разрешена компоновка всех массивов. ВЫП ПРОМЫ указывает на выполнение процедуры вывода информации на МЛ (по 5 символов).

Оператор РЗД служит для задания символов разделителей. В 17—21-й позициях указываются разделители в следующем порядке:

<конец зоны>, <конец связки>, <конец реквизита>, <конец признака связки>, <код зачеркивания>.

Если 17—21-я колонки пусты, то в качестве разделителей используются стандартные символы. Если РЗД отсутствует, то

* В скобки {} взято несколько значений параметра, одно из которых он принимает.

используются разделители, указанные в ОД (см. описание программы ОБРОД).

Оператор НЭИ служит для задания носителя исходной информации. Присутствие НЭИ обязательно. В случае ввода информации с ПЛ в 22-й позиции может стоять символ «З». В этом случае код зачеркивания обрабатывается как обычный символ.

Оператор ОПИСВ служит для описания вводимой с МЛ (по 5 символов) исходной информации. ОПИСВ может и отсутствовать. В этом случае ввод будет осуществляться с помощью описания массива, указанного в ОД.

Оператор ОПИСЫ аналогичен предыдущему, только служит для описания вывода исходной информации на МЛ (по 5 символов).

В программе предусмотрен контроль карт заказа на наличие ошибок. Все карты заказа выводятся на печать. Неверные карты, если они были обнаружены, помечаются кодом «*».

Обращение к ВВОДЛ через программу КООРДИНАТОР:

ВЫ — АААА; ПМ ◇ * АААА * ВВОДЛ00300ЛС

<количество листов МОЗУ>

<количество ЛПМ> ◇

Так как в программе ВВОДЛ предусмотрено динамическое распределение памяти, то, задавая различное количество листов МОЗУ, можно изменять длину поля, отведенного под ввод исходной информации. В начале работы программы допустимая длина перфоленты печатается на пишущей машинке.

Обращение к программе во внутреннем режиме. К программе предусмотрено обращение на ЯСК (работа во внутреннем режиме):

```
ИП   ВВОДЛ; 1
КА   А; В
```

Под этикеткой А сообщаются сведения, приведенные в табл. 3.4, под этикеткой В — в табл. 3.5.

Здесь ОН и ОК — соответственно начало и конец поля для ввода описаний;

ВВОДН и ВВОДК — начало и конец поля для ввода зоны исходной информации;

МАКЕТ и МАКЕК — то же для поля, в котором будут формироваться скомпонованные записи;

ПОЛН1 и ПОЛК1, ПОЛН2 и ПОЛК2 ПОЛН3 и ПОЛК3 — то же для полей обмена при выводе на магнитную ленту соответствующих компонуемых массивов;

БЛОКК и БЛОКМ — адреса блоков пользователя, подключаемых соответственно после контроля каждой связки и перед переносом скомпонованной записи в поле обмена;

Б1 и Б2 — адрес двух ячеек в блоке пользователя для передачи параметров программе пользователя.

Равенство нулю БЛОКК и Б1 (БЛОКМ и Б2) означает отсутствие соответствующего блока пользователя;

РЗД — адрес массива из пяти ячеек для передачи разделителей программе. Разделители находятся в нулевом символе каждой ячейки в порядке, аналогичном порядку в операторе СИМВО описаний. Если РЗД=0, то используются разделители, заданные в описании, или стандартные.

Во внутреннем режиме программа не освобождает и не закрепляет внешних устройств, не проверяет их готовности, не резервирует рабочих полей.

Если какой-либо из параметров в табл. 3.4 или 3.5 отсутствует, то вместо него обязательно должен быть код нуля.

Таблица 3.4

Этикетка	КОП	Адреса и замечания
А	КТ	<имя описаний>
	КА	ОН; ОК
	КА	ВВОДН; ВВОДК
	КА	МАКЕТ; МАКЕК
	КА	ПОЛН1; ПОЛК1
	КА	ПОЛН2; ПОЛК2
	КА	ПОЛН3; ПОЛК3
	КА	{О; БЛОКК Б1; БЛОКК}
	КА	{О; БЛОКМ Б2; БЛОКМ}
	КА	0; РЗД

Таблица 3.5

Этикетка	КОП	Адреса и замечания
В	КЧ	<управляющая ячейка>
	КНВУ	<для ввода ОД>
	КНВУ	<для ввода зоны ЭИ>
	КНВУ	<для вывода зоны ЭИ>
	КНВУ	<для вывода 1-го компонуемого массива>
	КНВУ	<для вывода 2-го компонуемого массива>
	КНВУ	<для вывода 3-го компонуемого массива>

Режим работы задается с помощью управляющей ячейки, которая имеет следующую структуру:

17-й разряд = 0 — описания находятся в МОЗУ (в поле, указанном пользователем). В этом случае содержимое 18-го разряда безразлично;

17-й разряд = 1 — описания находятся на МЛ или ПЛ;

18-й разряд = 1 — ввод описаний с МЛ;

18-й разряд = 0 — ввод описаний с ПЛ;

24-й разряд = 1 (0) — запрещен (разрешен) контроль структуры;

30-й разряд = 1 — зона исходной информации находится в МОЗУ, в этом случае происходит обычная работа без закрытия массивов;

30-й разряд = 0 — исходный массив находится на МЛ или ПЛ;

29-й разряд = 1 — то же, что и для 30-го разряда, но после обработки происходит закрытие массивов;

29-й разряд = 0 — после обработки не происходит закрытия массива.

С помощью 29 и 30 разрядов управляющей ячейки пользователь может сам вводить зоны исходного массива в оперативную память. В этом случае после ввода каждой очередной (не последней) зоны он должен предусмотреть обращение к программе, поместив предварительно в 30-й разряд 1, а после чтения последней зоны 30-й разряд управляющей ячейки должен иметь значение 0, а 29-й — 1;

31-й разряд = 1 — ввод зоны информации с ПЛ;

31-й разряд = 0 — ввод зоны информации с МЛ;

32-й разряд = 1 (0) — требуется (не требуется) выполнение процедуры «контроль зоны»;

33-й разряд = 1 (0) — нужен (не нужен) вывод на МЛ исходной информации;

34—36-й разряды означают разрешение компоновать те массивы, в соответствующих разрядах которых стоит 1.

В программе предусмотрена возможность при обращении вместо табл. 3.4 использовать табл. 3.6. В этом случае в 7-м разряде

Таблица 3.6

Этикетка	КОП	Адреса и замечания
А	КТ	<имя описаний>
	КА	$A_н$; $A_к$
	КА	О; БЛОКК Б1; БЛОК
	КА	О; БЛОКМ Б2; БЛОКМ
	КА	О; РЗД

управляющей ячейки должна быть 1. В табл. 3.6 $A_н$ и $A_к$ — соответственно начальный и конечный адреса области оперативной памяти, отведенной для размещения рабочих полей и полей обмена. Программа динамически распределит память этой области между полями на основании информации в машинных описаниях. Описания должны быть введены пользователем с адреса $A_н$, или это сделает сама программа в зависимости от значения 17-го разряда управляющей ячейки. После распределения оставшаяся часть области используется под ввод исходной информации, в этом случае ввод информации организуется только самой программой.

Аналогично вместо табл. 3.5 может использоваться табл. 3.7, в этом случае в 8-м разряде управляющей ячейки должна быть 1. Тогда программа ВВОДЛ сама резервирует внешние устройства, печатает сообщения об установке носителей.

Таблица 3.7

Этикетка	КОП	Адреса и замечания
В	КЧ	<управляющая ячейка>

В программе предусмотрена возможность подключения блоков пользователя в режиме внутренней программы только в следующих местах: а) после контроля очередной связки, если в ней не было обнаружено ошибок; б) перед переносом очередной скомпонованной записи в поле вывода.

При обращении к блоку пользователя программа передает следующие параметры. В случае а):

КА Д; ДН
КА О; <адрес обрабатываемой связки>

где Д — адрес массива из трех ячеек для текста ошибки пользователя. Если в Д остался код «О», то это означает, что ошибка пользователем обнаружена не была;

ДН — абсолютный адрес описания связки в ОД (описание связки, состоящее из семи ячеек).

В случае б):

КА О; ДН1
КА О; М,

где ДН1 — адрес (абсолютный) описания скомпонованной записи в ОД;

М — адрес (абсолютный) скомпонованной записи.

Блоки пользователя организуются двумя способами:

1) блок пользователя организован как отдельная программа, удовлетворяющая требованиям СМО ЭВМ «Минск-32». В этом случае в начале блока должны быть зарезервированы две ячей-

ки для приема параметров. Содержимое 8-й и 9-й ячеек табл. 3.4 при обращении к ВВОДЛ равно КА 0; БЛОКК, где БЛОКК — адрес описания блока пользователя;

Таблица 3.8

Этикетка	КОП	Адреса и замечания
	БАЗ	0
АННБП	НОП	
	РЗВ	2
	.	
	.	тело НБП
	.	
	ВЫХ	АННБП: 0

2) блок пользователя расположен во внешней программе и имеет структуру, приведенную в табл. 3.8. В этой таблице АННБП — адрес входа в НБП, указываемый в описании. В этом случае внутри (но не в начале) программы пользователя резервируются две ячейки для приема параметров, тогда содержимое 8-й и 9-й ячеек равно КА Б; БЛОК, где Б — адрес ячеек пользователя для приема параметров; БЛОК — абсолютный адрес блока пользователя.

Пример. Пусть имеется следующий входной документ:

Изделие		Деталь	
Шифр	Количество	Шифр	Количество
ИЗД1	10	ДЕТ1	2
		ДЕТ2	4
ИЗД2	100	ДЕ1	1
		ДЕ2	7

Связкам присвоим следующие признаки:

Имя связки	Признак связки
ИЗДЕЛИЕ	И
ДЕТАЛЬ	Д

Операционного листа для данного массива нет.

Описания записаны на магнитную ленту под именем КОНОД с помощью программы ОБРОД.

Документ перфорируется следующим образом:

```
***НБМАСС10000000000000000000000000*И)ИЗДI, 0;Д)ДЕТ1,2;ДЕТ2,4;
И)ИЗД2,100;Д)ДЕ1,1;ДЕ2,7;***КБМАСС 10000000000
```

Перфолента с данным документом обрабатывается с помощью программы ВВОДЛ. Ниже приведен пример из табл. 3.6 и 3.7 при обращении к программе во внутреннем режиме:

Этикетка	КОП	Адреса и замечания
А	КТ	КОНОД
	КА	АН; АН + 2000
	НОП	
	НОП	
	НОП	
В	КЧ	3000064В

Поле АН длиной в 2000 ячеек должно быть зарезервировано в программе пользователя.

3.3. ВВОД ЭКОНОМИЧЕСКОЙ ИНФОРМАЦИИ С ПЕРФОКАРТ

Рассмотрим универсальную программу* ввода информации с перфокарт. Она состоит из двух модулей: программы обработки описаний пользователя (ОБРАБ) и собственно программы ввода с перфокарт (ВВОДК).

Программа ОБРАБ осуществляет синтаксический контроль описаний, составленных пользователем, и приводит их к виду, удобному для работы основной программы.

Программа ВВОДК предназначена для ввода, контроля и компоновки массивов информации, подготовленных на перфокартах. Она позволяет производить при однократном обращении обработку исходных документов различной структуры, снабженных внутренним шифром. Каждый вводимый документ должен быть отперфорирован на отдельной карте. Реквизиты документов одной структуры должны быть расположены на карте в одних и тех же колонках. Если производится обработка документов различной структуры при однократном обращении к программе, то эти документы снабжаются внутренними шифрами, которые на всех информационных перфокартах располагаются в одних и тех же колонках, причем внутренний шифр должен содержать не более пяти символов.

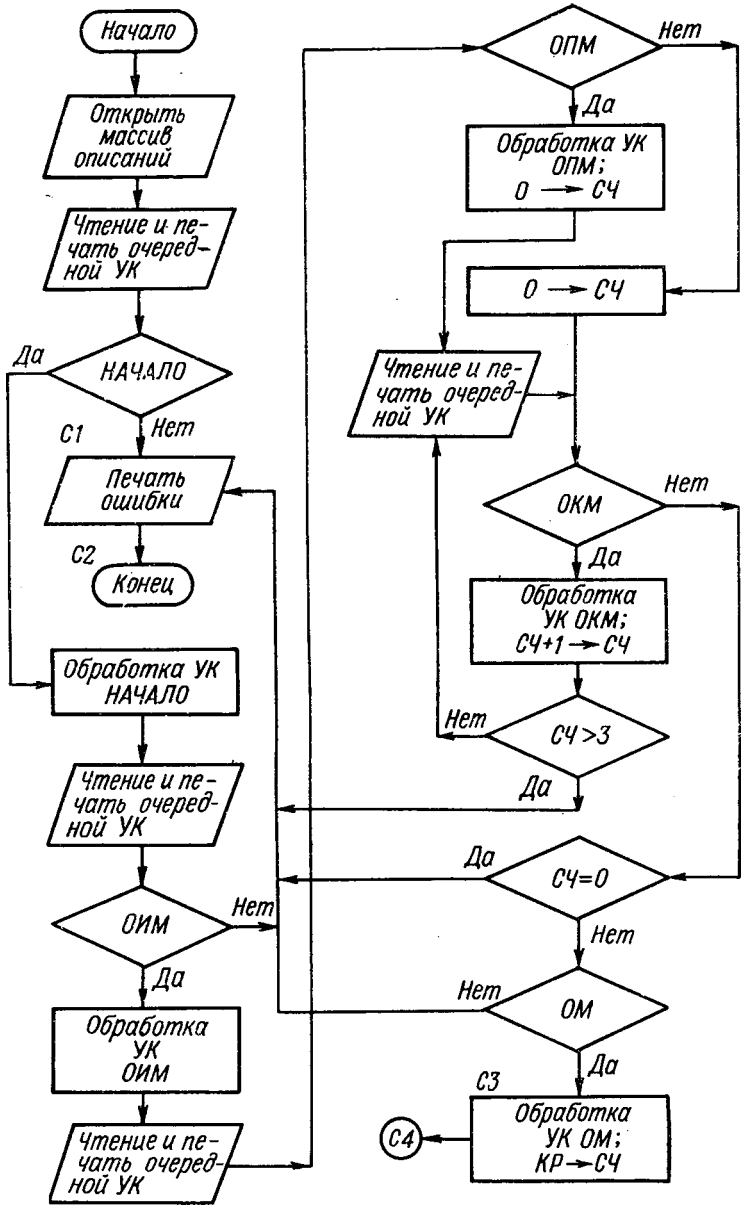
Программа ВВОДК в зависимости от описания объекта работы, составленного пользователем, а также заданного им режима работы в состоянии решать следующие задачи:

контролировать вводимую исходную информацию (ИИ);

выводить проверенную информацию на магнитные ленты (МЛ);

---

* В разработке программы принимали участие В. Н. Агафонов, В. А. Клишевская, В. М. Седенков, С. И. Таранко.



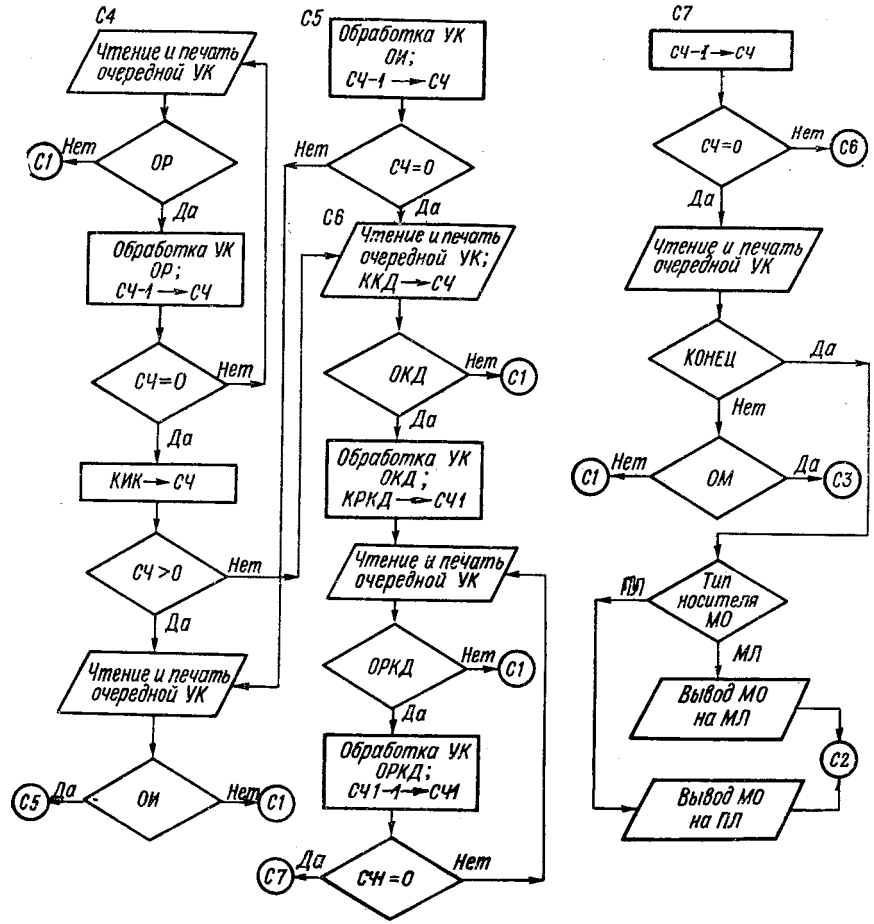


Рис. 3.8. Блок-схема программы ОБРАБ

проводить компоновку до трех массивов информации с последующим выводом на МЛ;

в процессе компоновки проводить редактирование исходной информации на уровне реквизитов.

Программа обработки описаний пользователя (ОБРАБ). Программа предназначена для переработки описаний, составленных пользователем, в вид, пригодный для дальнейшего использования (машинные описания) программой ввода экономической информации. Принципиальная блок-схема программы приведена на рис. 3.8. На ней используются следующие обозначения:

УК — управляющая карта;

КР — количество реквизитов;

КИК — количество итогов на карте;

ККД — количество компонуемых записей;

КРКД — количество реквизитов в компонуемых записях;

МО — машинные описания;

НАЧАЛО, ОИМ, ОПМ, ОКМ, ОМ, ОР, ОИ, ОКД, ОРКД, КОНЕЦ — наименования операторов в описании пользователя (см. ниже).

Описания в общем случае состоят из следующих составных частей: описания исходного массива, описания промежуточного массива, описаний компонуемых массивов, макетов исходных документов, реквизитов исходных документов, итогов на карте (в случае организации соответствующего контроля), компонуемых записей и реквизитов компонуемых записей.

Описания составляются на обычных бланках ССК. Каждый оператор описания записывается в отдельную строку бланка. При этом в позициях 12÷16 указывается имя оператора. Позиции 17÷21 служат для указания первого параметра, который представляет собой некоторый идентификатор (имя массива, шифр макета, имя записи и т. д.) и занимает всегда пять символов. Все остальные параметры разделяются запятыми и записываются в поле «Адреса и замечания», начиная с позиции 22. Признаком конца оператора является код пустой колонки (при перенесении на перфокарты). Поэтому свободные позиции при указании имени оператора, а также первого параметра (позиции 17÷21) заполняются пробелами. Первый символ имени оператора должен быть расположен в 12-й позиции.

Первым оператором описаний должен быть оператор НАЧАЛО, который имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
		НАЧАЛО, <ВН>

Здесь ВН — $\left\{\begin{matrix}\text{МЛ}\\\text{ПЛ}\end{matrix}\right\}$ — вид носителя машинных описаний.

Этот оператор является управляющим и указывает, на какой носитель нужно вывести массив машинных описаний — на магнитную ленту (МЛ) или на перфоленту (ПЛ).

Вслед за оператором НАЧАЛО следует оператор ОИМ, описывающий исходный массив и имеющий следующую структуру *:

Этикетка	КОП	Адреса и замечания
ОИМ	<ИМ>	[<ТК>][,ШМ=<НК>,<КК>]

Здесь ИМ — имя исходного массива;

ТК — тип карт $\left\{\begin{matrix}45\\80\end{matrix}\right\}$;

НК — начальная колонка шифра макета;

КК — конечная колонка шифра макета.

Оператор ОИМ присутствует всегда и является вторым оператором описания. Наличие имени массива в данном операторе обязательно. Остальные параметры либо присутствуют, либо нет. Если отсутствует параметр ТК, предполагается, что массив информации подготовлен на 80-колонных перфокартах.

Отсутствие параметра ШМ, указывающего местоположение шифра макета, означает, что описываемый для обработки массив состоит из документов одной структуры, не имеющих внутреннего шифра. Следует заметить, что если описываемый массив состоит из документов (макетов) одинаковой структуры, имеющих внутренний шифр, то программа ВВОДК осуществляет контроль на указанный шифр. В случае, если этот массив содержит разные по структуре макеты, внутренний шифр макета служит для поиска соответствующего ему описания.

Следующим оператором описания должен быть оператор ОПМ, описывающий промежуточный массив. Если пользователь не предусматривает вывод промежуточного массива, то оператор ОПМ должен отсутствовать.

Оператор ОПМ имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
ОПМ	<ИМ>	[МЛ=<ИЛ>[[,<срок годности>]

* Здесь и далее в скобки [] заключены параметры, которые могут быть опущены. Если в описании опущены значения параметров, идущих первыми, то следующие параметры сдвигаются на их место, при этом запятая перед первым параметром не ставится. Если опущены все параметры, то 22-я позиция должна быть пустой. В <> взяты идентификаторы параметров, на месте которых указывается одно из значений соответствующих параметров без этих кавычек. В скобки {} взято несколько значений параметра, одно из которых он принимает.

Здесь ИМ — имя промежуточного массива;

ИЛ — имя магнитной ленты для промежуточного массива.

Если в описании отсутствует имя магнитной ленты, то контроль МЛ, на которую выводится промежуточный массив, не производится.

Компонуемые массивы описываются с помощью операторов ОКМ, следующих за оператором ОПМ или ОИМ (при отсутствии оператора ОПМ). Количество операторов ОКМ соответствует количеству компонуемых массивов, но не должно быть больше трех. Имена компонуемых массивов должны быть различными.

Оператор ОКМ имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
ОКМ	<ИМ>	[<ФЗ>[<ДЗ>]] [,МЛ=<ИЛ>] [<срок годности>]

Здесь ИМ — имя компонуемого массива;

ФЗ — формат записи $\left\{\begin{matrix}\text{П}\\ \text{Ф}\end{matrix}\right\}$;

ДЗ — длина зоны;

ИЛ — имя МЛ для скомпонованного массива.

Если формат записи П, то вывод компонуемых записей на МЛ производится записями переменной длины; если Ф — записями фиксированной длины. Кроме формата записи, пользователь может указать длину информационной части зоны (ДЗ). Длина зоны задается десятичным числом. Например,

Этикетка	КОП	Адреса и замечания
ОКМ	МАССИ	Ф200, МЛ=АААААА,100

Согласно этому оператору массив компонуемых записей МАССИ выводится на МЛ с именем АААААА записями фиксированной длины. Длина выводимой зоны 200 ячеек, срок годности 100 дней. Если длина зоны не указана, то программой ВВОДК вывод будет производиться стандартными зонами длиной 320 ячеек. В этом случае пример оператора ОКМ будет иметь следующий вид:

Этикетка	КОП	Адреса и замечания
ОКМ	МАССИ	П,МЛ = АААААА,100

Если в операторе отсутствует формат записи (в этом случае должна отсутствовать и длина зоны), вывод на МЛ в программе ВВОДК будет производиться записями фиксированной длины и зонами стандартной длины.

На длины зон, указываемые пользователем, не накладывается никаких ограничений. Если в операторе ОКМ отсутствует имя МЛ, то в программе не будет осуществляться проверка на соответствие магнитных лент.

Срок годности в операторе ОКМ может отсутствовать. В этом случае предполагается, что срок хранения массива неограничен.

Макеты исходных документов описываются с помощью операторов ОМ. Вслед за операторами ОКМ должен следовать оператор ОМ для первого обрабатываемого макета.

Оператор ОМ имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
ОМ	<ШМ>	<КР>,<ККЗ> [,<КИК>] [,П=<ПР>, <КПКС>]

Здесь ШМ — шифр макета;

КР — количество реквизитов данного макета;

ККЗ — количество компонуемых записей;

КИК — количество итогов на карте (один символ);

ПР — символ, взятый пользователем в качестве признака карты суммирования (один символ);

КПКС — номер колонки, в которой помещается указанный признак на карте суммирования.

Наличие в операторе ОМ параметров ПР и КПКС указывает на то, что пользователь предусматривает контроль по карте суммирования. В этом случае наличие карты суммирования в обрабатываемом массиве обязательно, причем в отличие от информационных карт на карте суммирования должен быть помещен признак (ПР) в колонке КПКС.

Наличие параметра КИК указывает на необходимость контроля по итогам на карте.

Далее следуют операторы ОР, описывающие реквизиты этого макета. Количество операторов ОР должно соответствовать параметру КР, указанному в операторе ОМ.

Оператор ОР имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
ОР	<ШМ>	<ИР>, <ТРВ>, <НК>, <КК> [,С][<НГ> [,<ВГ]]

где ШМ — шифр макета, которому принадлежит данный реквизит;

ИР — имя реквизита;

ТРВ — тип реквизита входного документа $\left\{\begin{matrix} Т \\ Ц \\ В, <КЗПЗ> \end{matrix}\right\}$;

Т — текстовой;
Ц — целый;
В — вещественный;
КЗПЗ — количество знаков после запятой;
НК — номер начальной колонки реквизита;
КК — номер конечной колонки реквизита;
С — признак суммирования реквизита по столбцам;
НГ — число, определяющее нижнюю границу изменения реквизита;
ВГ — число, определяющее верхнюю границу изменения реквизита.

Наличие параметра С в операторе указывает на то, что данный реквизит участвует в суммировании. Параметры НГ и ВГ могут отсутствовать, в этом случае реквизиту программой присваиваются максимальная и минимальная границы. НГ и ВГ для текстового реквизита (Т) не имеют смысла и поэтому не задаются.

Вслед за операторами ОР в описании должны следовать операторы ОИ, описывающие итоги на карте. Количество таких операторов должно также соответствовать параметру КИК, указанному в операторе ОМ. Если в операторе ОМ параметр КИК отсутствует, т. е. пользователь не предусматривает контроля итога на карте для данного макета, то операторы ОИ должны отсутствовать. Количество итогов на карте не должно превышать трех.

Оператор ОИ, описывающий итог на карте, имеет структуру:

Этикетка	КОП	Адреса и замечания
ОИ	<ШМ>	<ИИР>, <ИСР>, <ИСР>. . .

где ШМ — шифр макета;
ИИР — имя реквизита итога на карте;
ИСР — имя суммируемого реквизита.

Суммируемых реквизитов по данному итогу может быть не более пяти.

Далее следуют описания компонуемых записей, производимых с помощью операторов ОКД.

Оператор ОКД имеет структуру:

Этикетка	КОП	Адреса и замечания
ОКД	<ШМ>	<ИКМ>, <ИЗ>, <КРКЗ>, <ДЛЗ>

где ШМ — шифр макета;

ИКМ — имя компонуемого массива, в который записывается данная запись;

ИЗ — имя записи;

КРКЗ — количество реквизитов в данной записи;

ДЛЗ — длина записи.

Из одного входного документа может компоноваться любое число различных по структуре записей, длина каждой не должна превышать 127 ячеек, при этом имена компонуемых записей должны быть различны.

Вслед за оператором ОКД должны следовать описания компонуемых в эту запись реквизитов (операторы ОРКД). Количество операторов ОРКД должно соответствовать параметру КРКЗ, указанному в операторе ОКД.

Оператор ОРКД имеет структуру:

Этикетка	КОП	Адреса и замечания
ОРКД	<ИЗ>	<ИР>, <ТР>, <НЯ>, <НР>, <ДЛР>

где ИЗ — имя записи, в которой помещается данный реквизит;

ИР — имя реквизита, компонуемого в запись;

ТР — тип реквизита.

Тип реквизита в компонуемой записи может быть одним из следующих: Т, ДЦ, ПЛ, ВФ, ДФ, ВБ, ВЗ, где Т — текстовый; ДЦ — десятичный целый; ПЛ — двоичный с плавающей запятой; ВФ — восьмеричный с фиксированной запятой; ДФ — десятичный с фиксированной запятой; ВБ — восьмеричный целый без знака; ВЗ — восьмеричный целый со знаком (первый разряд, выделенный под реквизит, используется как знаковый).

НЯ — номер начальной ячейки реквизита.

В зависимости от типа реквизита параметры НР и ДЛР специфичны.

Для реквизитов типа Т:

НР — номер начального символа;

ДЛР — количество символов, которое отводится под реквизит в записи.

Для реквизитов типа ДЦ:

НР — номер бита, с которого начинается первая тетрада;

ДЛР — количество тетрад.

Реквизиты типа Т или ДЦ могут занимать всю ячейку, часть ее либо несколько ячеек. Если тип реквизита ПЛ, ВФ или ДФ, то параметры НР и ДЛР не задаются и предполагается, что реквизит занимает всю ячейку.

Для реквизитов типа ВБ и ВЗ:

НР — номер начального бита;

ДЛР — количество битов.

В случае, если в качестве реквизита в компонуемую запись необходимо поместить литерал, оператор ОРКД имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
ОРКД	<ИЗ>	(<Л>), <НЯ>, <НС>, <КС>

где ИЗ — имя записи;
Л — литерал;
НЯ — номер ячейки;
НС — номер символа, начиная с которого в ячейку записи помещается литерал;
КС — количество символов, отведенное под литерал. Длина литерала не должна превышать пяти символов.

Количество описанных компонуемых записей (операторы ОКД) должно соответствовать параметру ККЗ, указанному в операторе ОМ. Описанием компонуемых записей завершаются описания, связанные с первым макетом. Далее следуют операторы, касающиеся очередного макета, начиная с оператора ОМ и т. д. Описания завершаются указанием всех операторов, касающихся последнего макета.

Последним оператором описаний является оператор КОНЕЦ, указывающий на конец описаний и имеющий следующую структуру:

Этикетка	КОП	Адреса и замечания
		КОНЕЦ

Каждый оператор описания перфорируется на отдельную карту. Массив описаний оформляется по всем правилам организации массивов СМО ЭВМ «Минск-32».

В процессе обработки описания пользователя подвергаются синтаксическому и семантическому контролю. В каждом операторе проверяется последовательность указания параметров, корректность их представления. Кроме того, контролируется порядок следования операторов и их число.

Для примера рассмотрим следующий оператор описания макета:

Этикетка	КОП	Адреса и замечания
ОМ	ЗАПАС	15,2,3,П—*,79

Здесь описан макет, состоящий из 15 реквизитов. Из данного макета компонуются две логические записи. Кроме того, по макету предусматривается три контроля по итогу на карте. Поэтому вслед за данным оператором должны следовать 15 операторов ОР, далее 3 оператора ОИ, и далее 2 оператора ОКД. Вслед за каждым из операторов ОКД должны следовать операторы ОРКД. Их число должно соответствовать параметру КРКЗ в операторе ОКД.

В программе ОБРАБ предусмотрена проверка соответствия имен реквизитов (операторы ОИ, ОРКД) именам реквизитов исходного макета (операторы ОР); имен компонуемых массивов (операторы ОКД) именам массивов, указанных в операторах ОКМ, и т. д.

В процессе обработки описания пользователя выводятся на АЦПУ с указанием соответствующих ошибок. Если ошибок не обнаружено, полученные машинные описания выводятся на магнитную ленту или перфоленту (согласно указанию в операторе НАЧАЛО). Вызов программы осуществляется с помощью программы КООРДИНАТОР (режим внешней программы), обращение имеет следующий вид:

ВЫ — АААААА; ПМ◇*АААААА*ОБРАБ00100 $\left\{\begin{matrix}\text{ЛС}\\\text{МЛ}\end{matrix}\right\}$ 01201◇.

Пример. В качестве примера приведем описание массива экономической информации, состоящего из документов одинаковой структуры (одного макета). Массив подготовлен на 80-колонных перфокартах и имеет имя МАС1. Для шифра макета на карте отведены 1-я и 2-я колонки.

Для записи проконтролированного массива используется магнитная лента ТОМ1. Имя промежуточного массива МАССИ.

Документ имеет следующую структуру:

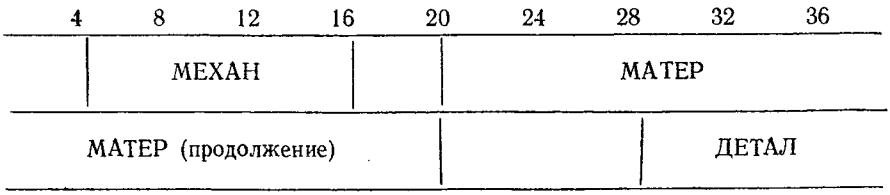
Шифр макета	Шифр механизма	Шифр технологической операции	Шифр детали	Расход			Материал	Количество
				1-й квартал	2-й квартал	1-ое полугодие		

Все реквизиты в данном массиве должны подчиняться следующим условиям:

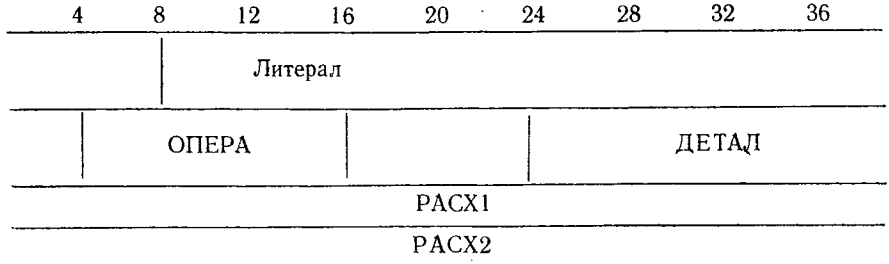
Полное наименование реквизита	Имя реквизита	Колонки, занимаемые реквизитом (НК-КК)	Тип реквизита (ТРВ)	Нижняя граница (НГ)	Верхняя граница (ВГ)	Признак суммирования
Шифр механизма	МЕХАН	3—5	Ц	17	777	—
Шифр технологической операции	ОПЕРА	6—9	Ц	2	9999	—
Шифр детали	ДЕТАЛ	10—11	Ц	5	87	—
Расход за 1-й квартал	РАСХ1	12—13	В,1	0,5	58	—
Расход за 2-й квартал	РАСХ2	14—15	В,1	0,5	58	—
Расход за 1-е полугодие	РАСХО	16—18	В,1	1	99	С
Материал	МАТЕР	19—23	Т	—	—	—
Количество	КОЛИЧ	24—26	Ц	13	665	С

На основании данного массива МАС1 нужно получить массив МАС2 на магнитной ленте (имя — ТОМ2). Новый массив будет состоять из записей двух типов:

запись с именем ДОК1



запись с именем ДОК2:



Реквизиты записи ДОК1 удовлетворяют условиям:

Полное наименование реквизита	Имя реквизита	Тип реквизита (ТР)	Номер ячейки (НЯ)	Начальный разряд (символ) (НР)	Длина реквизита (ДЛР)
Шифр механизма	МЕХАН	ДЦ	0	5	3
Материал	МАТЕР	Т	0	3	5
Шифр детали	ДЕТАЛ	ВБ	1	28	9
Расход на 1-е полугодие	РАСХО	ДФ	2	—	—
Количество	КОЛИЧ	ПЛ	3	—	—

реквизиты записи ДОК2 удовлетворяют условиям:

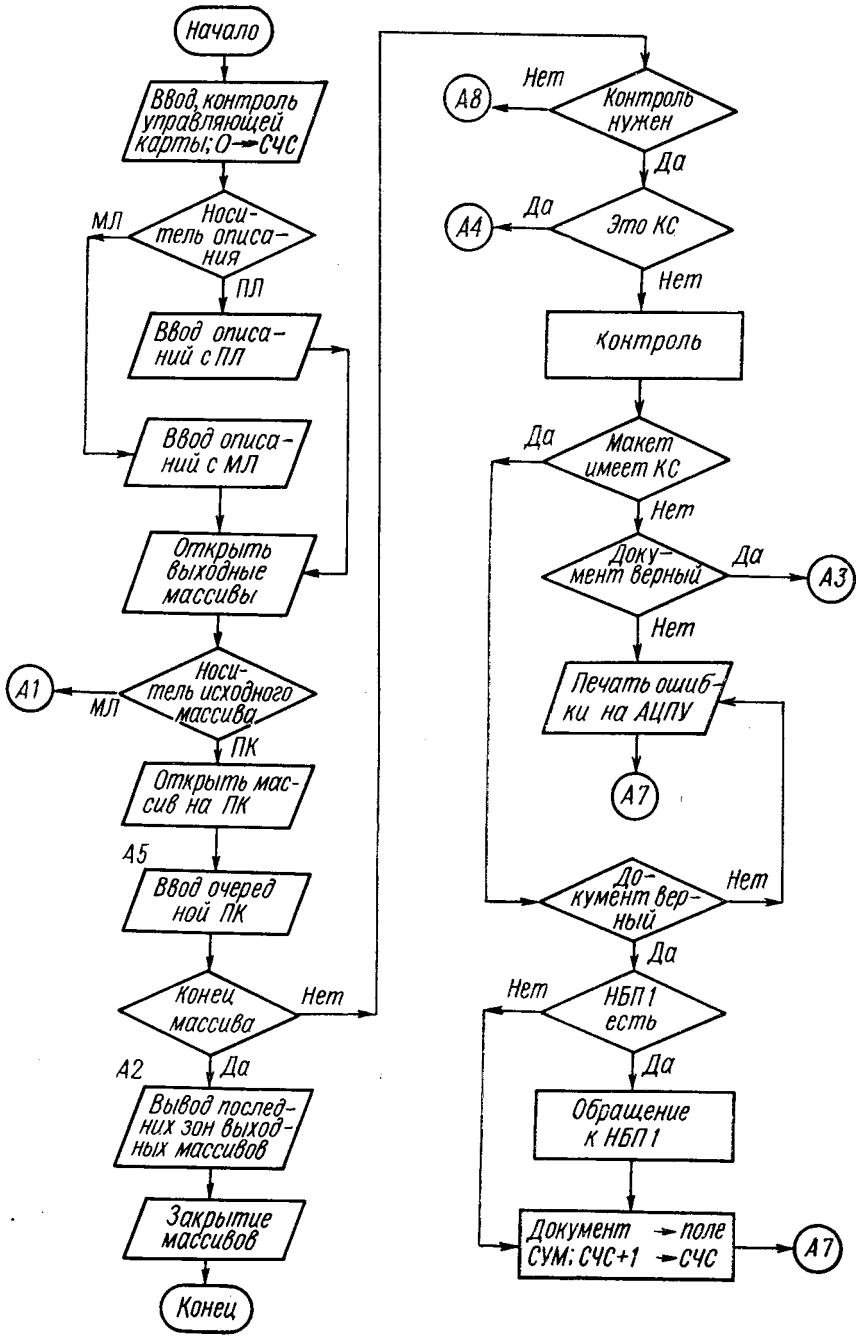
Полное наименование реквизита	Имя реквизита	Тип реквизита (ТР)	Номер ячейки (НЯ)	Начальный разряд (символ) (НР)	Длина реквизита (ДЛР)
Литерал	ИЮНЬ		0	1	4
Шифр технологической операции	ОПЕРА	ВЗ	1	4	12
Шифр детали	ДЕТАЛ	ДЦ	1	25	3
Расход за 1-й квартал	РАСХ1	ПЛ	2	—	—
Расход за 2-й квартал	РАСХ2	ПЛ	3	—	—

Операторы описания для данного массива информации будут представлены на бланке ССК следующим образом:

Этикетка	КОП	Адреса и замечания
		НАЧАЛО, МЛ
ОИМ	МАС1	80, ШМ=1,2
ОПМ	МАССИ	МЛ-ТОМ1, 100
ОКМ	МА С2	Ф, МЛ—ТОМ2
ОМ	ОЗ	8, 2, 1, П-Е, 27
ОР	ОЗ	МЕХАН, Ц, 3 , 5. 17, 777
ОР	ОЗ	ОПЕРА, Ц, 6, 9, 2, 9999
ОР	ОЗ	ДЕТАЛ, Ц, 10, 11, 5, 87
ОР	ОЗ	РАСХ1, В, 1, 12, 13, 0. 5, 58
ОР	ОЗ	РАСХ2,В, 1, 14, 15, 0. 5, 58
ОР	ОЗ	РАСХО, В, 1, 16, 18, С, 1, 99
ОР	ОЗ	МАТЕР, Т, 19, 23
ОР	ОЗ	КОЛИЧ, Ц, 24, 26, С, 13, 665
ОИ	ОЗ	РАСХО, РАСХ1,РАСХ2
ОКД	ОЗ	МАС2, ДОК1, 5, 4
ОРКД	ДОК1	МЕХАН, ДЦ, 0, 5, 3
ОРКД	ДОК1	МАТЕР, Т, 0, 3, 5
ОРКД	ДОК1	ДЕТАЛ, ВБ, 1, 28, 9
ОРКД	ДОК1	РАСХО, ДФ, 2
ОРКД	ДОК1	КОЛИЧ, ПЛ, 3
ОКД	ОЗ	МАС2, ДОК2, 5, 4
ОРКД	ДОК2	(июнь), 0, 1, 4
ОРКД	ДОК2	ОПЕРА, ВЗ, 1, 4, 12
ОРКД	ДОК2	ДЕТАЛ, ДЦ, 1, 25, 3
ОРКД	ДОК2	РАСХ1, ПЛ, 2
ОРКД	ДОК2	РАСХ2, ПЛ, 3
		КОНЕЦ

В результате работы программы ОБРАБ эти описания, переработанные в машинный вид, будут выведены на магнитную ленту, имя которой пользователь укажет с пишущей машинки.

Программа ввода с перфокарт (ВВОДК). Программа предназначена для ввода, контроля и компоновки массивов информации с исходных одноуровневых документов, в качестве промежуточного носителя используются перфокарты. Данная программа для своей работы использует машинные описания, полученные программой обработки описаний пользователя.



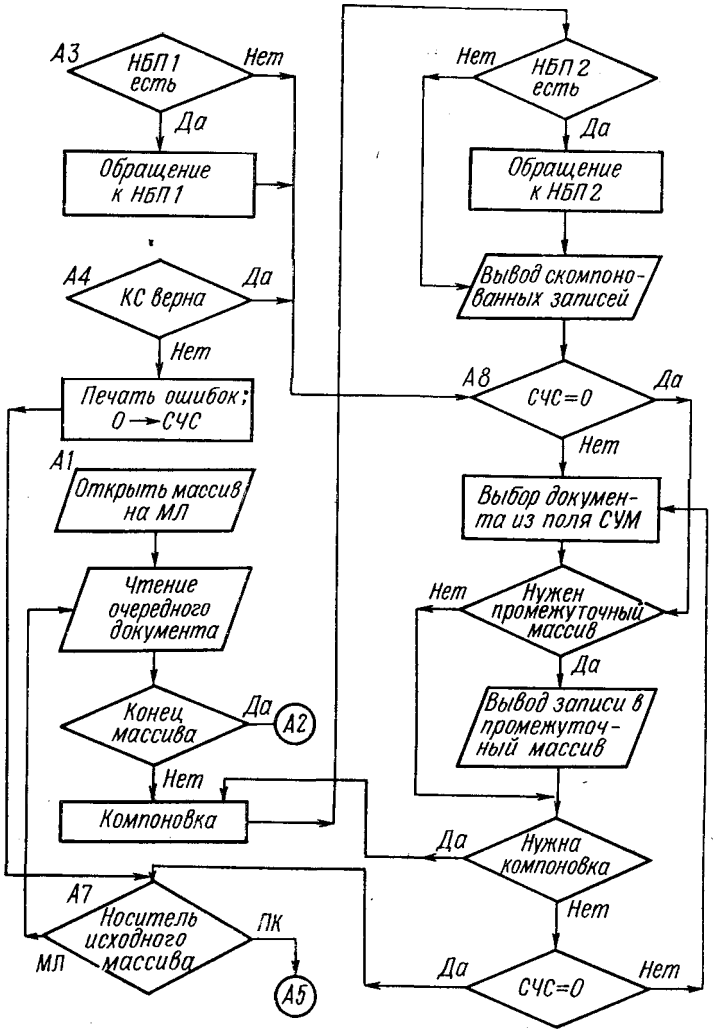


Рис. 3.9. Блок-схема программы ВВОДК

Блок-схема программы ВВОДК приведена на рис. 3.9. На ней используются следующие обозначения:

КС — карта суммирования;

НБП1 — нестандартный блок пользователя, выполняемый после контроля;

НБП2 — нестандартный блок пользователя, выполняемый после компановки;

поле СУМ — буфер для накопления пачки карт исходного массива;

СЧС — счетчик количества карт в пачке.

В программе предусмотрены следующие режимы работы:

— ввод и контроль информации с перфокарт с выводом на магнитную ленту без компоновки;
— ввод и контроль информации с перфокарт с выводом на магнитную ленту с компоновкой;
— ввод и контроль информации с перфокарт без вывода на магнитную ленту и без компоновки;
— ввод и контроль информации с перфокарт без вывода на магнитную ленту с компоновкой;
— ввод и компоновка информации с магнитной ленты без контроля;
— ввод и компоновка информации с перфокарт без контроля.

Массив входной информации может состоять из разнотипных документов. Документы, предназначенные для обработки этой программой, перфорируются на перфокарты в коде ГОСТ 10859—64. Предполагается, что каждый реквизит занимает жестко определенные колонки и на каждой карте перфорируется один документ. Массив исходных документов должен быть подготовлен к работе по правилам организации массивов на перфокартах СМО ЭВМ «Минск-32».

При организации контроля по карте суммирования (КС) массив информационных карт должен быть разбит на пачки карт одинакового макета. Последней картой пачки должна быть карта суммирования. При установке карт на устройство ввода с перфокарт указанные пачки не должны разрываться. Однако сразу могут устанавливаться несколько пачек.

При перфорации количественных реквизитов, имеющих значащие цифры после запятой, запятая не перфорируется. В этом случае на каждой карте должно быть отперфорировано одно и то же количество знаков после запятой. Место запятой указывается в описании соответствующего реквизита (параметр ТР оператора ОР). В отличие от информационных карт карта суммирования должна содержать в определенной колонке признак карты суммирования. Шифр макета и результаты суммирования по столбцам нужных реквизитов пробиваются в тех же колонках, что и на информационной карте.

Технологический процесс обработки информации данной программой состоит в общем случае из следующих этапов:

— ввода и контроля исходного массива;
— вывода проконтролированного массива на МЛ;
— компоновки информации в записи заданной структуры;
— вывода массива скомпонованных записей на МЛ;
— выдачи на АЦПУ сообщений об ошибках.

Алгоритм программы осуществляет следующие виды контроля исходной информации: по итоговому реквизиту на карте, для чего пользователь составляет описание итога на карте (операторы ОИ); по карте суммирования, которая находится в массиве информационных карт и представляет собой сумму по столбцам некоторых реквизитов впереди стоящих карт (по требованию пользователя); на соответствие верхней и нижней границам реквизита (по требованию пользователя); на соответствие шифров макетов, указанных в описании, с шифрами в документе; по типу реквизита.

Обнаруженные неверные документы печатаются с соответствующим признаком и исключаются из дальнейшей обработки.

Контроль по итоговому реквизиту на карте заключается в суммировании по абсолютной величине указанных пользователем суммируемых реквизитов и сравнении результата суммирования с итоговым реквизитом по модулю его разрядности.

При организации контроля по карте суммирования суммирование реквизитов информационных карт, имеющих итог на карте суммирования, производится по модулю. В случае если результат суммирования не вмещается в отведенные для реквизита колонки, то старшие разряды сумм опускаются. В случае если результаты суммирования не совпадают с соответствующими данными на карте суммирования, то из дальнейшей обработки исключается вся пачка.

Сообщения об ошибках выводятся на АЦПУ по следующей форме:

Тип ошибки	№-ПК	ШМ	Информация

В графе №-ПК печатается номер перфокарты; в графе ШМ — шифр соответствующего макета; в графе Информация — реквизиты исходного документа в той последовательности, которая дана в описании (реквизиты разделяются пробелами).

Алгоритм программ построен таким образом, что в результате однократного выполнения программы в одном из вышеперечисленных режимов осуществляется полная обработка одного входного документа. Если в результате контроля исходной информации обнаружена ошибка, неверный документ, содержащий ошибки, выдается на АЦПУ; одновременно печатается сообщение об ошибке и управление передается на обработку следующего документа, а неверный документ не включается в выходной массив.

В программе предусмотрена возможность подключения нестандартных блоков пользователя в режиме внутренней программы в следующих местах: 1) после контроля очередного документа, если в нем не было обнаружено ошибок; 2) перед переносом очередной скомпонованной записи в поле вывода. В первом случае пользователь имеет возможность провести дополнительный контроль, во втором — некоторые преобразования полученной скомпонованной записи.

Результатом работы программы является образование не более трех массивов, выведенных на различные магнитные ленты. Различные по структуре компонуемые записи, каждая из которых имеет постоянную для своего типа длину, могут по желанию пользователя выводиться в один или разные массивы записями переменной или фиксированной длины. Если в один массив компонуются различные по структуре записи, то их длины должны быть равны при выводе записями фиксированной длины и могут быть различными при выводе записями переменной длины. Вывод производится зонами фиксированной длины. Длина зоны для каждого массива задается пользователем.

Таблица 3.9

Номер колонки	Значение	Пояснения
1		
2—3	{ПК / МЛ}	Вид носителя исходной информации
4	{ 1 / 0 }	Признак — нужен контроль (1), нет (0)
5	{ 1 / 0 }	Признак — нужна запись на МЛ проконтролированной информации (1), нет (0)
6	{ 1 / 0 }	Признак — первый из описанных массивов нужно компоновать и записывать на выходную МЛ (1), нет (0)
7	{ 1 / 0 }	Аналогично для второго массива
8	{ 1 / 0 }	Аналогично для третьего массива
9—10	{ПЛ / МЛ}	Вид носителя машинных описаний
11—15	<XXXXX>	Имя массива машинных описаний на ПЛ или МЛ
16—20	<AAAAA>	Имя магнитной ленты, если массив машинных описаний находится на МЛ, или — пусто

Вызов программы может осуществляться через КООРДИНАТОР (режим внешней программы), обращение имеет следующий вид:

ВЫ—АААА; ПМ◇*АААА*ВВОДК<время> $\left\{\begin{matrix}\text{МЛ}\\\text{ЛС}\end{matrix}\right\}$ <количество листов МОЗУ> <количество ЛПМ>◇

В этом режиме запуск программы для выполнения определенных функций осуществляется с помощью управляющей карты, оформленной как массив.

Структура управляющей карты приведена в табл. 3.9.

Ниже приведен пример структуры одной из управляющих карт:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
	П	К	1	1	0	1	1	М	Л	О	П	И	С	А	К	К	К	К	К		...

Согласно данной управляющей карте, исходный массив расположен на перфокартах, предусмотрен контроль входной информации, запись проконтролированного массива на МЛ (создание промежуточного массива), компоновка второго и третьего массивов выходных записей. Описания, используемые программой и полученные программой ОБРАБ, находятся на магнитной ленте ККККК и имеют имя ОПИСА.

Управляющая карта оформляется как массив. Имя массива управляющей карты в начальном и конечном блоках — ЗАКАЗ. В программе предусмотрен соответствующий контроль управляющей карты. В случае, если будет обнаружена ошибка, оператору выдается сообщение — НЕВЕРЕН ЗАКАЗ.

Программа динамически распределяет память, заданную пользователем, на основании информации из управляющей карты.

Таблица 3.10

Этикетка	КОП	Адреса и замечания
Н	КЧ	<управляющая ячейка>
	КТ	<имя ОД>
	КА	$\left\{\begin{matrix}\text{О; БЛОКК}\\\text{Б1; БЛОКК}\end{matrix}\right\}$
	КА	$\left\{\begin{matrix}\text{О; БЛОКМ}\\\text{Б2; БЛОКМ}\end{matrix}\right\}$
	КА	ЗОНА1; ЗОНА1+К1
	КА	ЗОНА2; ЗОНА2+К2
	КА	ЗОНА3; ЗОНА3+К3
	КА	АС; АС+К4

Кроме того, к программе предусмотрено обращение на ЯСК (работа во внутреннем режиме):

ИП ВВОДК;1
КА М;Н

здесь М — адрес выхода по сбою, Н — этикетка таблицы информации, приведенной на табл. 3.10, где

имя ОД — имя массива описаний исходной информации;

БЛОКК и БЛОКМ — адреса блоков пользователя, подключаемых соответственно после контроля каждого документа и перед переносом скомпонованной записи в поле обмена;

Б1 и Б2 — адреса двух ячеек для передачи параметров блоку пользователя;

ЗОНА1 и ЗОНА1+К1, ЗОНА2 и ЗОНА2+К2, ЗОНА3 и ЗОНА3+К3 — соответственно начало и конец полей обмена при выводе на магнитную ленту соответствующих компонуемых массивов;

АС и АС+К4 — начало и конец рабочего поля, отведенного под суммирование. Резервирование этих полей осуществляется пользователем в программе обращения.

Блоки пользователя могут организовываться двумя способами: 1) блок пользователя организован как отдельная программа, удовлетворяющая требования СМО ЭВМ «Минск-32». В этом случае в начале блока должны быть зарезервированы две ячейки для приема параметров. Содержимое 3-й и 4-й ячеек табл. 3.10 при обращении к ВВОДК соответственно равны:

КА О;БЛОКК
КА О;БЛОКМ

2) блок пользователя расположен во внешней программе и имеет структуру, приведенную в табл. 3.11, где тело НБП — текст программы пользователя, внутри которой резервируются две

Таблица 3.11

Этикетка	КОП	Адреса и замечания
	БАЗ	0
{ БЛОКК / БЛОКМ }	НОП	
	РЗВ	2
	{ . / . / . }	<тело НБП>
	ВЫХ	{ БЛОКК / БЛОКМ } ;0

ячейки для приема параметров. В этом случае содержимое 3-й, 4-й ячеек табл. 3.10 равно

КА Б1;БЛОКК
КА Б2;БЛОКМ

При обращении к блоку пользователя программа передает следующие параметры. В случае 1)

КА ОШ;Д
КА О;ДН

где ОШ — адрес массива из пяти ячеек, первая из которых отводится под признак ошибки, следующие четыре — под тип ошибки;

Д — абсолютный адрес описания макета;

ДН — адрес обрабатываемого документа.

В случае 2)

КА О;Д1
КА О;ДН1

где Д1 — абсолютный адрес описания компонуемой записи;

ДН1 — абсолютный адрес скомпонованной записи.

Структура ячейки признака ошибки следующая:

0-й символ = { 1 — карта суммирования; 0 — информационная карта;

1-й символ = { 1 — в результате контроля в данном документе была обнаружена ошибка; 0 — документ верный;

2-й символ = { 1 — в блоке суммирование не проводилось; 0 — суммирование проводилось.

Если какой-либо из параметров в табл. 3.10 отсутствует, то вместо него обязательно должен быть код нуля. Кроме того, в программе предусмотрена возможность использования сокращенной таблицы информации (табл. 3.12).

В этом случае АН и АК — соответственно начальный и конечный адреса области, отведенной для размещения рабочих

Таблица 3.12

Этикетка	КОП	Адреса и замечания
Н	КЧ	<управляющая ячейка>
	КТ	<имя ОД>
	КА	{ О; БЛОКК / Б1; БЛОКК }
	КА	{ О; БЛОКМ / Б2; БЛОКМ }
	КА	АН; АК

полей и полей обмена. Программа динамически распределит память этой области между полями на основании информации машинных описаний, т. е. при необходимости компоновать какой-либо из массивов программа отведет соответствующий участок области для поля обмена. В случае необходимости оставшаяся часть области используется программой в качестве рабочего поля суммирования. На пишущую машинку выдается сообщение о максимальном количестве карт в пачке, которые одновременно могут участвовать в суммировании.

Режим работы программы (при обращении на ЯСК) задается с помощью управляющей ячейки, которая имеет следующую структуру:

7-й разряд = { 1, если при обращении к программе задана сокращенная таблица информации;
0 — полная таблица информации;

18-й разряд = { 1 — ввод описаний с МЛ;
0 — ввод описаний с ПЛ;

24-й разряд = { 1 — запрещено суммирование;
0 — разрешено предусмотренное суммирование;

31-й разряд = { 1 — ввод информации с перфокарт;
0 — ввод информации с МЛ;

32-й разряд = { 1 — нужен контроль;
0 — не нужен контроль;

33-й разряд = { 1 — нужен вывод промежуточного массива на МЛ;
0 — вывод не нужен;

34—36-й разряды — означают разрешение компоновать те массивы, в соответствующих разрядах которых стоит 1.

Для примера рассмотрим выполнение программы ВВОДК для ввода массива, описанного в примере к программе ОБРАБ. В этом случае пользователю необходимо подготовить две магнитные ленты с именами ТОМ1 и ТОМ2 и на указываемые программой ЛПМ. Массив описаний в данном случае устанавливается на ЛПМ.

Управляющая карта в режиме отдельной программы будет иметь вид:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	П	К	1	1	0	1	0	М	Л	М	О	П	И	Д	Т	Т	Т	Т	Т	...

ТТТТТ — имя МЛ.

В результате выполнения программы на магнитной ленте ТОМ1 будет получен промежуточный массив, который может использоваться в дальнейшей работе. На ленте ТОМ2 будет получен выходной массив.

3.4. ПРОГРАММА СОРТИРОВКИ ИНФОРМАЦИИ НА МАГНИТНЫХ ЛЕНТАХ

Программа * предназначена для сортировки большого массива информации, размещенного на одной магнитной ленте. Упорядочение массива логических записей выполняется в возрастающей или убывающей последовательности с учетом знаков ключевых реквизитов или по их абсолютной величине.

В программе используется метод полифазного слияния [44]. Данный метод сортировки предполагает при своей работе два этапа: внутренняя сортировка порций информации и их предварительное распределение; слияние упорядоченных порций в единый массив.

На первом этапе исходная информация, находящаяся на магнитной ленте пользователя, распределяется на две рабочие магнитные ленты таким образом, чтобы количества порций записей на них являлись членами последовательности чисел Фибоначчи, которая определяется следующим образом:

$$F_0=0;$$
$$F_1=1;$$
$$F_i=F_{i-2}+F_{i-1}, \text{ для } i\geqslant 2.$$

Распределение информации происходит по следующему алгоритму:

а) организуются счетчики для подсчета порций, записываемых на рабочие ленты (d_1, d_2 — счетчики соответственно для рабочих лент МЛ1 и МЛ2), для чего $d_1=0$, $d_2=0$;

б) задаются характеристические числа распределения этапа $x_1=F_0$, $x_2=F_1$ (x_1, x_2 — требуемые числа порций записей на лентах МЛ1, МЛ2 для данного этапа распределения);

в) с исходной магнитной ленты считывается зонами информация до заполнения выделенного для сортировки поля ОП. Полученная порция сортируется в оперативной памяти по методу Шелла и записывается на одну из рабочих МЛ.

На ленту МЛ1 записывается x_1-d_1 порций, затем на ленту МЛ2 — x_2-d_2 порций. При каждой записи порции на ленту МЛ1 или МЛ2 к счетчикам d_1 или d_2 соответственно прибавляется 1. В результате процесс записи на ленту МЛ1 или МЛ2 происходит до тех пор, пока x_1-d_1 или x_2-d_2 соответственно не станут равными нулю. Процесс может прекратиться, если информация на исходной МЛ исчерпана. Если информация на исходной магнитной ленте еще не исчерпана, то переходим к следующему этапу распределения. Для этого вычисляются новые значения характеристических чисел распределения: новое значение x_1 равно старому значению x_2, новое значение x_2 равно сумме старых значений x_1 и x_2. И процессы чтения информации с исходной

* В разработке программы принимали участие Ю. А. Акулов, В. С. Обоницкий.

МЛ, внутренняя сортировка и запись на одну из рабочих МЛ продолжаются (возвращаемся на повторение подпункта в)).

Данное распределение происходит до тех пор, пока вся исходная информация не будет считана с магнитной ленты и распределена на две рабочие ленты.

В результате предварительного распределения могут иметь место три случая. Количества порций на магнитных лентах МЛ1 и МЛ2 совпадают со значениями характеристических чисел последнего этапа распределения ($d_1=x_1$, $d_2=x_2$), или запись порций окончилась, не достигнув требуемого распределения на лентах МЛ1 и МЛ2 ($d_1<x_1$, $d_2<x_2$), или на ленте МЛ1 записано $d_1=x_1$ порций, а на ленте МЛ2 не достигнуто значение характеристического числа ($d_2<x_2$).

Обозначим через Φ_1 и Φ_2 числа порций, недостающих соответственно на магнитных лентах МЛ1 и МЛ2 до характеристических чисел x_1, x_2, т. е.

$$\left.\begin{aligned}\Phi_1 &= x_1 - d_1;\\ \Phi_2 &= x_2 - d_2.\end{aligned}\right\} \qquad (1)$$

Будем называть недостающие до требуемого распределения порции на лентах фиктивными.

Этап слияния использует три ленты: МЛ1 и МЛ2 с информацией и свободную ленту МЛ3, которую будем называть лентой вывода. Он осуществляется по следующему алгоритму:

а) если $x_1 \neq 0$, вычисляем количество фиктивных порций на лентах МЛ1 и МЛ2 по формуле (1), если же $x_1 = 0$, то уже получили рассортированный массив;

б) если $k = \Phi_2 - \Phi_1 \neq 0$, то пересылаем k порций с МЛ1 на ленту вывода МЛ3 и переходим к пункту в); если $k = 0$, то

в) производим слияние порций с двух лент на ленту вывода, пока одна из лент не освободится. Освободившаяся лента будет лентой вывода на следующем этапе слияния;

г) вычисляем значения счетчиков и характеристических чисел по следующему алгоритму:

пересылаем d_1 в d_2 и x_1 в R;
вычисляем $x_2 - x_1$ и записываем в d_1;
пересылаем x_1 в x_2 и d_1 в x_1;

д) производим переименование магнитных лент: МЛ1 назовем МЛ3, МЛ2—МЛ1, МЛ3—МЛ2 и возвращаемся к пункту а).

Блок-схема программы сортировки приведена на рис. 3.10.

В программу сортировки включены также процедуры «проверка рассортированности массива» и «дублирование массива на МЛ пользователя». Эти процедуры по желанию пользователя могут быть включены в работу после процесса сортировки как вместе, так и отдельно друг от друга. В процессе работы процедуры «проверка рассортированности массива», если встречаются не упорядоченные логические записи, выдается на пишущую

машинку (ПМ) пульта оператора содержимое первых ячеек этих записей.

Заметим, что промежуточные массивы, выходной массив и его копия записываются зонами, длина которых равна длине зоны исходного массива. Копия массива получает имя исходно-

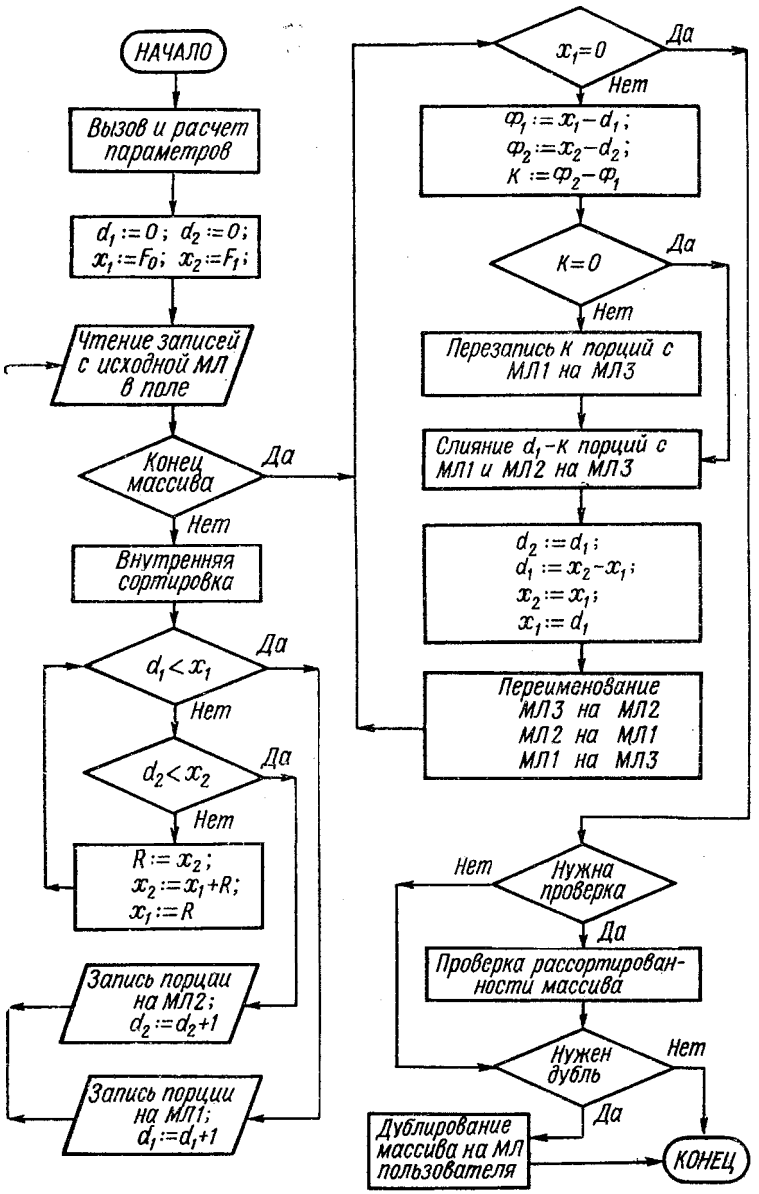


Рис. 3.10. Блок-схема программы МСОРТ

го массива, однако пользователь может изменить имя копии рассортированного массива. Для этого достаточно сообщить новое имя после соответствующего указания на ПМ.

Организация массивов информации, обрабатываемых программой, должна отвечать требованиям СМО ЭВМ «Минск-32». Эти массивы должны состоять из записей фиксированной длины и записываться на магнитную ленту зонами постоянного объема, причем последняя зона может быть укороченной. Программа сортировки позволяет упорядочивать записи входного массива в соответствии с одним или несколькими ключевыми реквизитами внутри каждой записи. Ключевые реквизиты могут быть и числовыми, и буквенными, а также занимать часть ячейки, целую ячейку или размещаться в нескольких ячейках логической записи. Числовые ключевые реквизиты могут быть положительными и отрицательными и быть представлены в любом из следующих форматов: восьмеричным или десятичным числом, числом с фиксированной или с плавающей запятой.

Результатом работы программы является упорядоченный исходный массив, записанный на одной из рабочих магнитных лент. Номер НМЛ, на котором находится МЛ с выходным массивом, сообщается оператору на пишущую машинку. По желанию пользователя, кроме того, может быть получена копия результатной информации.

Программа в процессе работы выдает на пишущую машинку пульта сообщение о ходе сортировки, а также о различных сбойных ситуациях.

Обращение к программе сортировки имеет следующий вид:

Этикетка	КОП	Адреса и замечания
	ИП	МСОРТ; 3
	КА	АН; АК
	КА	АТРК; АТИ
	КА	А; Д

где

Этикетка	КОП	Адреса и замечания
МСОРТ	ОПР	МСОРТ
	НОП	

АН — начальный адрес резервируемой пользователем области оперативной памяти (ОП);

АК — конечный адрес резервируемой пользователем области ОП без учета 9 ячеек (длина резервируемой пользователем об-

ласти ДЛ=3×К×ДЗ+9, где К=1, 2, 3, ...; ДЗ — длина зоны исходного массива);

АТРК — начальный адрес описания ключевых реквизитов;

АТИ — начальный адрес таблицы информации;

А — начальный адрес резервируемой области ОП, длина которой равна количеству ключевых реквизитов;

Д — начальный адрес описания массива.

Описание ключевых реквизитов имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
АТРК	КИ	<КРЕКВ>; <ДЛЗ>
	КЧ	±<НЯ><РН><РК> <ПЗ><МД>
. . .	. .	. . . . .

где КРЕКВ — количество ключевых реквизитов;

ДЛЗ — длина записи.

Описания ключевых реквизитов должны располагаться сплошным массивом в порядке убывания их значимости (приоритета при сортировке). Значимость ключевых реквизитов определяется пользователем.

Описание ключевого реквизита занимает одну ячейку. Знак ячейки описания ключевого реквизита определяет, по возрастанию или убыванию значения соответствующего реквизита производится сортировка (плюс — по возрастанию, минус — по убыванию).

Параметры НЯ, РН, РК задаются в восьмеричной системе счисления.

НЯ — номер ячейки реквизита в логической записи (разряды 1÷15);

РН — номер левого бита реквизита (без учета знакового) в указанной записи, причем $0<РН\leqslant 44$ (разряды 16÷21);

РК — номер правого бита реквизита в указанной записи, причем $0<РК\leqslant 44$ (разряды 22÷27);

ПЗ — 0 или 1 (разряды 31÷33). Если ПЗ=1, то сравнение реквизитов производится с помощью команд арифметики с плавающей запятой; если ПЗ=0, то — с помощью команд арифметики с фиксированной запятой;

МД — 0, или 1, или 3 (разряды 34÷36). Если МД=0, сравнение реквизитов производится по модулю. Если реквизит в логической записи занимает целую ячейку или часть ее и сравнение надо производить с учетом знака, то МД=1. Если же реквизит в логической записи занимает более одной ячейки и сравнение надо производить с учетом знака, тогда реквизит описывается поячеечно, причем для первой ячейки реквизита МД=1, а для следующих ячеек МД=3.

Таблица информации имеет следующий вид:

Этикетка	КОП	Адреса и замечания
АТИ	КТ	ИМЯ1
	КТ	ИМЯ2
	КЧ	ПРОВ

где ИМЯ1 — имя МЛ с исходным массивом (5 символов);

ИМЯ2 — имя МЛ для записи рассортированного массива, если его нужно дублировать с рабочей МЛ на ленту пользователя (5 символов). Если дублировать рассортированный массив не требуется, ИМЯ2=0;

ПРОВ=0 или 1. Если ПРОВ=1, то производится проверка рассортированности выходного массива. Если ПРОВ=0, то проверка не производится.

Описание исходного массива, требуемое для программы сортировки, составляется согласно требованиям СМО ЭВМ «Минск-32» [23].

Время работы программы (Т) зависит главным образом от длины исходного массива и резервируемого объема оперативной памяти. Если $F_{i-1} < \left[\frac{N}{v}\right] \leqslant F_i$, тогда время выполнения сортировки можно определить по формуле

$$T = \frac{N}{v} t_{ш} + 1{,}2 \cdot 10^{-3} v \sum_{j=3}^{i} F_{i-j+1} \cdot F_j \approx \frac{N}{v} t_{ш} + 2{,}3 \cdot 10^{-3} v\,(i - 2) \cdot F_{i-2},$$

где F_i — i-е число последовательности чисел Фибоначчи;

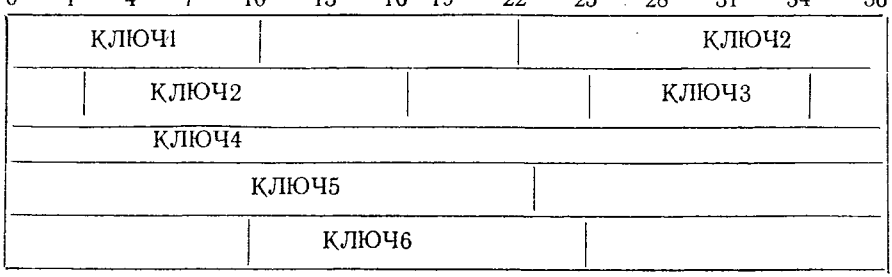
N — длина исходного массива (в словах);

v — объем памяти, выделенный для сортировки (в словах);

$t_{ш}$ — время сортировки первоначальной порции методом Шелла, *сек*;

T — время работы программы, *сек*.

Пример. Пусть требуется рассортировать массив «МАСС1», записанный на МЛ с именем «ЛЕНТ1». Массив записан зонами длиной 400 ячеек (без учета трех контрольных ячеек). Запись этого массива имеет следующую структуру:



Сортировку будем производить по возрастанию числовых значений реквизитов КЛЮЧ2, КЛЮЧ1, КЛЮЧ4 с учетом знака и по убыванию реквизитов КЛЮЧ5, КЛЮЧ3, КЛЮЧ6. Знак должен находиться в первом разряде соответствующих реквизитов. Причем КЛЮЧ4 представлен числом с плавающей запятой, КЛЮЧ6 не имеет знака, а КЛЮЧ5 представлен текстовой информацией.

В конце процесса сортировки требуется также провести проверку на рассортированность массива и сделать его копию на МЛ с именем «↑ЛЕНТ».

Тогда для этого случая обращение к программе будет иметь вид:

Этикетка	КОП	Адреса и замечания
	ИП	СОРТ; 3
	КА	ПОЛЕ; ПОЛЕ+1200
	КА	АТРК; АТИ
	КА	А; Д

где

Этикетка	КОП	Адреса и замечания
СОРТ	ОПР	МСОРТ
	НОП	
АТРК	КИ	7; 5
	КЧ	+2744001В
	КЧ	+10117003В
	КЧ	+111001В
	КЧ	+20144011В
	КЧ	—30125000В
	КЧ	—13240000В
	КЧ	—41230000В
АТИ	КТ	ЛЕНТ1
	КТ	ЛЕНТ
	КЧ	1
Д	КТ	МАСС1
	НОП	
	КЧ	—0
	НОП	
	КНВУ	МЛ, 1
	КЧ	400
	НОП	
	НОП	
А	РЗВ	7
ПОЛЕ	РЗВ	1209

3.5. ПРОГРАММА ВНУТРЕННЕЙ СОРТИРОВКИ ИНФОРМАЦИИ

Программа * предназначена для упорядочения массива записей фиксированной длины в возрастающей или в убывающей последовательности по одному или нескольким ключевым реквизитам, произвольно расположенным внутри каждой записи, с учетом знака этих ключевых реквизитов или по их абсолютной

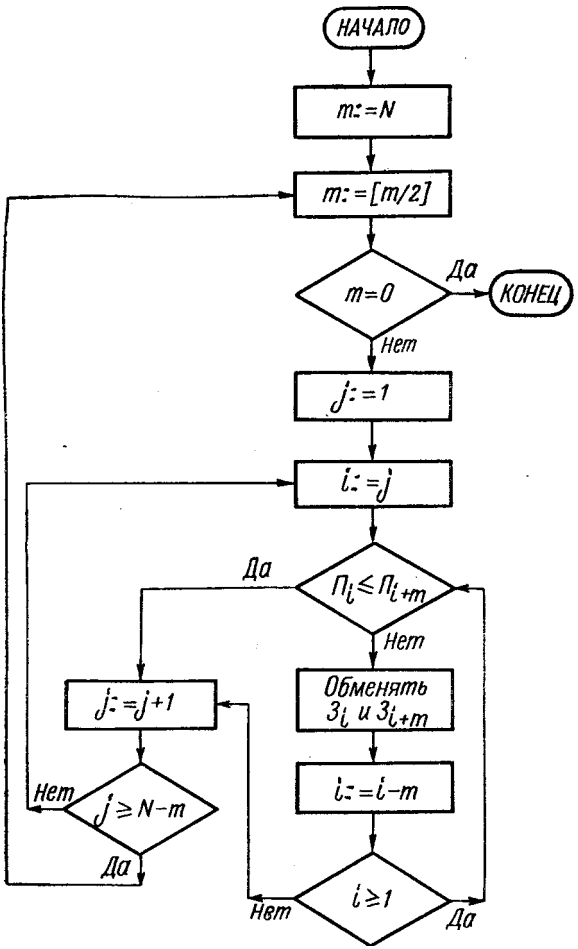


Рис. 3.11. Блок-схема программы СОРТШ

величине. В программе использован известный метод Шелла [3].

Блок-схема программы приведена на рис. 3.11. В ней использованы следующие обозначения:

---

* В разработке программы принимали участие Ю. А. Акулов, В. С. Обоницкий.

N — количество записей;
$З_i$ — i-я запись;
$П_i$ — ключевые реквизиты i-й записи;
$\left\lfloor \frac{m}{2} \right\rfloor$ — целая часть числа $\frac{m}{2}$;
m, i, j — рабочие ячейки.

Программа позволяет упорядочивать массив записей фиксированной длины, расположенный в оперативной памяти. Ключевые реквизиты записей могут быть цифровыми, алфавитными и алфавитно-цифровыми и располагаться в любом месте записи. Числовые ключевые реквизиты могут иметь знак и быть представленными в любом из следующих форматов: восьмеричным или десятичным числом, числом с фиксированной или с плавающей запятой. Массив должен содержать целое число записей.

Обращение к данной программе имеет следующий вид:

Этикетка	КОП	Адреса и замечания
	ИП	СОРТШ; 2
	КА	АН; АК+1
	КА	А; АТРК

где

Этикетка	КОП	Адреса и замечания
СОРТШ	ОПР	СОРТШ
	НОП	

АН — начальный адрес массива в ОП;
АК — конечный адрес массива в ОП;
А — начальный адрес резервируемой пользователем области ОП, длина которой равна количеству ключевых реквизитов;
АТРК — начальный адрес описания ключевых реквизитов.

Описание ключевых реквизитов имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
АТРК	КИ	<КРЕКВ>; <ДЛЗ>
	КЧ	±<НЯ><РН><РК> <ПЗ><МД>
	. .	. . . . .

Все параметры, которые задаются в этом описании, должны быть представлены восьмеричными числами.

В первой строке этого описания задаются: КРЕКВ — количество ключевых реквизитов; ДЛЗ — длина записи.

Каждая следующая строка представляет собой описание ключевого реквизита. Знак числа определяет порядок упорядочения (плюс — по возрастанию, минус — по убыванию).

НЯ — номер ячейки реквизита в логической записи (разряды 1÷15);

РН — номер левого бита реквизита (без учета знакового) в указанной записи, причем $0 \leqslant РН < 44$ (разряды 16÷21);

РК — номер правого бита реквизита в указанной записи, причем $0 < РК \leqslant 44$ (разряды 22÷27);

ПЗ — 0 или 1 (разряд 33). Если ПЗ=1, то сравнение реквизитов производится с помощью команд арифметики с плавающей запятой; если ПЗ=0, то — с помощью команд арифметики с фиксированной запятой;

МД — 0, или 1, или 3 (разряды 35÷36). Если МД=0, то сравнение реквизитов производится по модулю. Если реквизит в логической записи занимает целую ячейку или часть ее и сравнение надо производить с учетом знака, тогда МД=1. Если же реквизит в логической записи занимает более одной ячейки и сравнение надо произвести с учетом знака, тогда реквизит описывается поячеечно, причем для первой ячейки реквизита МД=1, а для следующих — МД=3.

Заметим, что описания ключевых реквизитов должны располагаться сплошным массивом в порядке убывания их значимости (приоритета при сортировке).

Пример. Пусть требуется рассортировать массив, структура записей которого приведена в примере для программы МСОРТ. Пусть массив содержит 120 записей и расположен в ОП в поле с начальным адресом АН. Массив должен быть рассортирован по убыванию реквизита КЛЮЧ4 с учетом знака и по возрастанию абсолютного значения реквизита КЛЮЧ3.

Тогда для этого случая обращение к программе будет иметь вид:

Этикетка	КОП	Адреса и замечания
	ИП	Б; 2
	КА	АН; АН+600
	КА	А; АТРК
Б	ОПР	СОРТШ
	НОП	
АТРК	КИ	2; 5
	КЧ	—20144011В
	КЧ	+13240000В
А	РЗВ	2

3.6. ПРОГРАММА КОРРЕКТИРОВКИ ЭКОНОМИЧЕСКОЙ ИНФОРМАЦИИ НА МАГНИТНЫХ ЛЕНТАХ

Программа * предназначена для внесения изменений в корректируемый массив (старый справочник) согласно информации корректирующего массива (корректуры). При этом предполагается, что старый справочник и корректура имеют записи постоянной длины и одинаковым образом упорядочены соответственно по возрастанию или по убыванию ключевых реквизитов. Отметим, что записи старого справочника и корректуры могут иметь различную структуру и разную длину. В таком случае подключением нестандартных блоков пользователя (НБП) структура этих записей должна приводиться к структуре записи, которая описана в обращении.

Ключевые реквизиты могут быть представлены в любом из форматов, предусмотренных в программах ввода экономической информации с перфоленты и перфокарт (см. разделы 3.2 и 3.3). Суммируемые реквизиты (основания) также могут быть представлены в любом из форматов за исключением Т (текстовый).

Результатом работы программы является новый справочник, полученный по правилам, описанным ниже. Если в процессе работы программы один из массивов (старый справочник или корректура) окажется не рассортированным, программа после соответствующего сообщения на ПМ пульта оператора прекращает работу и передает управление указанному в обращении адресу.

Программа может работать в одном из следующих трех режимов (при наличии массива корректур):

— вставка-удаление-замена (ВУЗК);
— подсуммирование (СУМК);
— слияние (СЛНК).

Корректировка осуществляется последовательным просмотром старого справочника и корректуры и формированием нового справочника на основании известного принципа «отец-сын» [11].

Блок-схемы программы приведены на рис. 3.12 и 3.13. На них использованы следующие обозначения: ЗС — запись массива старого справочника; ЗК — запись массива корректур; N — количество старших реквизитов, отличных от нуля в ЗК на групповое удаление.

При работе в режиме ВУЗК производится последовательная подчистка записей старого справочника и корректуры, т. е. в формировании нового справочника участвуют только последние записи из группы записей с одинаковыми ключевыми реквизитами. Далее программа над полученными вышеуказанным

---

* В разработке программы принимали участие В. Н. Агафонов, Ю. А. Акулов, П. И. Грибко, В. С. Обоницкий.

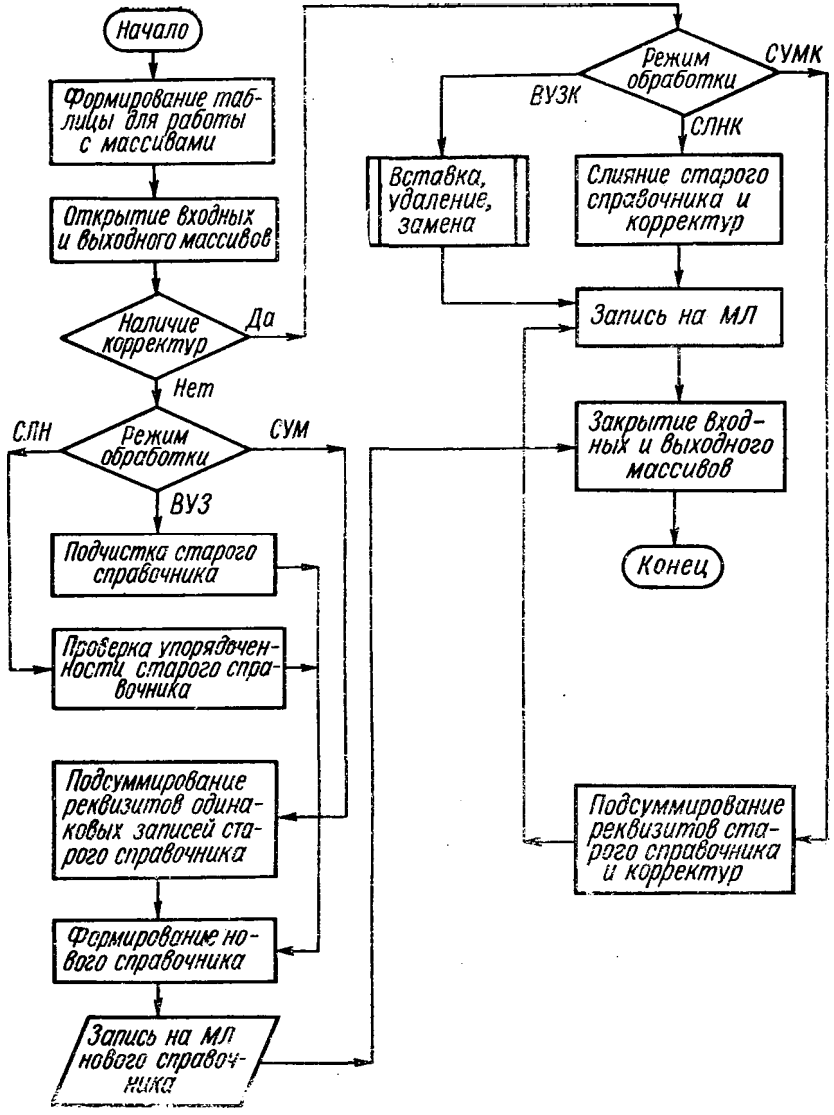


Рис. 3.12. Блок-схема программы КОРЕК

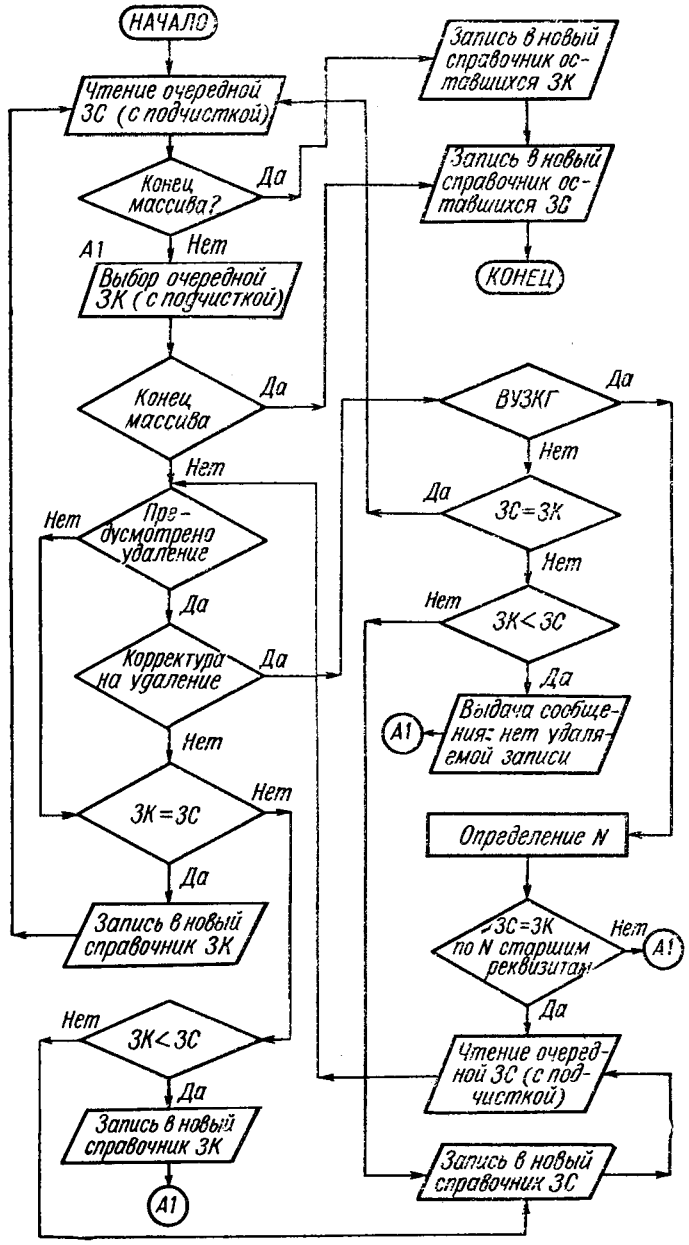


Рис. 3.13. Блок-схема подпрограммы «Вставка, удаление, замена»

способом записями позволяет выполнить следующие функции:
— вставку записи корректуры;
— замену записи старого справочника записью корректуры;
— удаление записи старого справочника.

Вставка записи корректуры в новый справочник производится при несовпадении ее ключевых реквизитов ни с одной записью старого справочника и если эта запись не указывает на необходимость удаления.

Если ключевые реквизиты старого справочника и корректуры совпадают и корректура не указывает на необходимость удаления, то в новый справочник попадает запись корректуры (замена).

Запись корректуры указывает на необходимость удаления при совпадении значения какого-то реквизита (реквизитов) с определенным значением (в частности, с нулем). Признак удаления описывается пользователем. Ни одна запись старого справочника, которая совпадает по ключевым реквизитам с одной из корректур на удаление, не попадает в новый справочник (удаление).

В программе предусмотрена возможность группового удаления записей старого справочника (режим ВУЗКГ), у которых значения старших ключевых реквизитов одинаковы с соответствующими значениями ключевых реквизитов корректуры на удаление. При этом сравнение ведется до первого нулевого значения ключевого реквизита корректуры. Например, если записи справочника содержат четыре ключевых реквизита «шифр цеха», «шифр участка», «шифр детали», «шифр операции» и необходимо удалить из массива все логические записи по заданному цеху и участку, то для этого достаточно в массиве корректур иметь логическую запись на удаление, в которой приводятся эти значения реквизитов «шифр цеха» и «шифр участка», а значения реквизитов «шифр детали» и «шифр операции» должны быть нулевыми. При использовании режима ВУЗКГ следует помнить, что если некоторые ключевые реквизиты могут принимать значения, равные нулю, то пользоваться режимом ВУЗКГ нельзя, так как обычная корректура на удаление, в которой эти реквизиты окажутся нулевыми, будет рассматриваться как корректура на групповое удаление.

При работе программы в режиме СУМК производится подсуммирование по описанным пользователем реквизитам-основаниям логических записей старого справочника и соответственно корректуры, имеющих одинаковые значения ключевых реквизитов, т. е. в корректировке участвуют подсуммированные записи старого справочника и корректуры.

Если подсуммированные записи старого справочника и корректуры совпадают по значениям ключевых реквизитов, то в новый справочник помещается запись, полученная сложением этих записей по тем же реквизитам-основаниям.

Несовпадающие по ключевым реквизитам подсуммированные записи старого справочника и корректуры также помещаются в новый справочник.

При работе программы в режиме СЛНК в новый справочник помещаются логические записи старого справочника и корректуры. Причем при совпадении значений ключевых реквизитов логических записей старого справочника и корректуры первыми заносятся записи старого справочника.

Во всех описанных режимах производится проверка массива старого справочника и корректуры на корректность сортировки. При обнаружении нерассортированности выдается сообщение оператору и программа прекращает работу.

В программе предусмотрены следующие режимы работы с одним массивом:

— ВУЗ — подчистка массива;
— СУМ — подсуммирование массива;
— СЛН — проверка на корректность сортировки массива и получение дубля.

Следует заметить, что режим ВУЗК[Г] особенно удобен при корректировке нормативно-справочной информации. Однако он может оказаться полезным при корректировке промежуточных массивов. Режим СУМ[К] особенно полезен при организации накапливаемых массивов оперативной информации. Режим СЛНК может использоваться для реализации процедуры дозаписи и особенно полезен для организации включения некоторых оперативных данных в основной массив.

Обращение к программе (на ЯСК) имеет следующий вид:

Этикетка	КОП	Адреса и замечания
	ИП	КОРЕК; 1
	КА	А; ОКОР
КОРЕК	ОПР	КОРЕК
	НОП	

где ОКОР — адрес описаний объектов работы;

А — адрес выхода из программы КОРЕК в случае нерассортированности массива старого справочника или корректуры.

Описание объектов работы имеет следующий вид:

Этикетка	КОП	Адреса и замечания
ОКОР	КТ	<РЕЖР>
	КА	<ДЛЗ>; АОКР
	КА	АОСР; АОПУ
	КА	АОМС; АТАС

Этикетка	КОП	Адреса и замечания
	КА	АОМК; АТАК
	КА	АОМН; АТАН
	КА	АППОМ; АППЗМ
	КА	АППЧС; АППЧК
	КА	АППСР; АППЗН

где <РЕЖР> = $\left\{\begin{matrix}\text{ВУЗ [К [Г]]}\\ \text{СУМ [К]}\\ \text{СЛН [К]}\end{matrix}\right\}$ — режим работы программы;

ДЛЗ — длина логической записи, обрабатываемой программой КОРЕК (в восьмеричной системе счисления);

АОКР — начальный адрес описания ключевых реквизитов записи;

АОСР — начальный адрес описания суммируемых реквизитов (задается только для режимов СУМ и СУМК, в других случаях АОСР=0);

АОПУ — начальный адрес описания признака удаления (задается только для режимов ВУЗК или ВУЗКГ, в других случаях АОПУ=0);

АОМС — начальный адрес описания массива старого справочника;

АТАС — начальный адрес таблицы адресов, определяющих поле ввода и поля для записей старого справочника;

АОМК — начальный адрес описания массива корректур (задается для всех режимов, кроме ВУЗ, СУМ и СЛН, в этом случае АОМК=0);

АТАК — начальный адрес таблицы адресов, определяющих поле ввода и поля для записей корректур (для режимов ВУЗ, СУМ и СЛН АТАК=0);

АОМН — начальный адрес описания массива нового справочника;

АТАН — начальный адрес таблицы адресов, определяющих поле вывода и поле для записи нового справочника;

АППОМ — начальный адрес нестандартного блока пользователя (НБП), выполняемого после открытия массивов старого справочника и корректур;

АППЗМ — начальный адрес НБП, выполняемого перед закрытием нового справочника;

АППЧС — начальный адрес НБП, выполняемого после чтения очередной записи старого справочника;

АППЧК — начальный адрес НБП, выполняемого после чтения очередной записи корректуры (для режимов ВУЗ и СУМ АППЧК=0);

АППСР — начальный адрес НБП, выполняемого перед очередным сравнением записи старого справочника и корректуры;

АППЗН — начальный адрес НБП, выполняемого перед выводом очередной записи в новый справочник.

Признаком отсутствия любой НБП служит нулевой адрес в соответствующем месте описания.

Описания ключевых реквизитов имеют следующую структуру:

Этикетка	КОП	Адреса и замечания
АОКР	КЧ	<КРЕКВ>
	КТ	<ТИП> { В / У }
	КЧ	<НЯ><НР> <ДЛНР> В
	. .	. . . . .

где КРЕКВ — количество ключевых реквизитов. Для каждого ключевого реквизита составляется двухъячеечное описание, в котором

ТИП — тип реквизита, указываемый двухсимвольным кодом:

Т␣ — текстовый (используется только для АОКР);

ВБ — восьмеричный без знака;

ВЗ — восьмеричный со знаком;

ВФ — восьмеричный фиксированный;

ДЦ — десятичный целый;

ДЗ — десятичный со знаком;

ДФ — десятичный фиксированный;

ПЛ — плавающий;

{В / У} — порядок рассортированности массива, если В — по возрастанию, если У — по убыванию данного ключевого реквизита;

НЯ — относительный адрес реквизита в записи (разряды 1 ÷ 18);

НР — начальный разряд реквизита (разряды 19 ÷ 24);

ДЛНР — длина реквизита в разрядах (разряды 25 ÷ 36) (НЯ, НР, ДЛНР задаются в восьмеричной системе; В — указание восьмеричной константы). Для форматов ВФ, ДФ и ПЛ: НР = 0, ДЛНР = 0. Для формата Т НР означает номер символа, а ДЛНР — количество символов в реквизите.

Таблица описания суммируемых реквизитов имеет следующую структуру:

Этикетка	КОП	Адреса и замечания
АОСР	КЧ	<КРЕКВ1>
	КТ	<ТИП>
	КЧ	<НЯ><НР> <ДЛНР> В

Описания суммируемых реквизитов аналогичны описаниям ключевых реквизитов, только в типе реквизита отсутствует текстовый формат (Т) и в первой ячейке описания любого реквизита не указывается порядок рассортирования массива.

Таблица описания признака удаления имеет следующий вид:

Этикетка	КОП	Адреса и замечания
АОПУ	КИ	НЯПУ; ДЛПУ
	КЧ ... КЧ	<Трафарет>
	КЧ ... КЧ	<Признак удаления>

где НЯПУ — относительный адрес ячейки логической записи, с которой размещается признак удаления;

ДЛПУ — длина признака удаления (в ячейках), начиная с относительного адреса НЯПУ, фактически ДЛПУ — длина трафарета.

Трафарет составляется по длине признака удаления и представляет собой константу выделения признака удаления.

Признак удаления представляет собой значения, по которым определяется корректура на удаление. Трафарет и признак удаления занимают точно ДЛПУ ячеек.

Таблицы описания полей ввода-вывода и полей для записей имеют следующую структуру:

Этикетка	КОП	Адреса и замечания
АТАС	КА	АНПВС; АКПВС
	КА	АНЗС1; АКЗС1
	КА	АНЗС2; АКЗС2
АТАК	КА	АНПВК; АКПВК
	КА	АНЗК1; АКЗК1

Этикетка	КОП	Адреса и замечания
	КА	АНЗК2; АКЗК2
АТАН	КА	АНПЫН; АКПЫН
	КА	АНЗН; АКЗН

где АНПВС (АНПВК), АКПВС (АКПВК) — соответственно начальный и конечный адреса поля ввода старого справочника (корректуры);

АНЗС1 (АНЗК1), АНЗС2 (АНЗК2) — начальный адрес соответственно первого и второго полей для записи старого справочника (корректуры);

АКЗС1 (АКЗК1), АКЗС2 (АКЗК2) — конечный адрес соответственно первого и второго полей для записи старого справочника (корректуры);

АНПЫН, АКПЫН — соответственно начальный и конечный адреса поля вывода нового справочника.

АНЗН, АКЗН — соответственно начальный и конечный адреса поля записи нового справочника.

Длины полей ввода старого справочника и корректуры должны быть кратны длинам зон соответствующих массивов с учетом трех контрольных ячеек.

Пользователь может задавать любую длину поля вывода, кратную длине записи нового справочника, с учетом трех контрольных ячеек.

Следует заметить, что программой допускается обработка различных по структуре записей старого справочника и корректуры и получение нового справочника, имеющего записи любой структуры. В этом случае пользователь обязан с помощью соответствующих НБП приводить записи старого справочника и корректуры к одной структуре, описанной в обращении к программе КОРЕК, а перед выводом записи в новый справочник преобразовать ее в нужную ему структуру. В этом случае длина полей записи должна выбираться согласно следующим правилам:

$$АКЗС1 - АНЗС1 = АКЗС2 - АНЗС2 = \max\{ДЛЗ,\ ДЗС\} + 2;$$
$$АКЗК1 - АНЗК1 = АКЗК2 - АНЗК2 = \max\{ДЛЗ,\ ДЗК\} + 2;$$
$$АКЗН - АНЗН = \max\{ДЛЗ,\ ДЗН\} + 2;$$

где ДЛЗ — длина описанной логической записи; ДЗС, ДЗК, ДЗН — длины логических записей старого справочника, корректуры и нового справочника соответственно.

Следует иметь в виду, что в описаниях массивов старого справочника, корректуры и нового справочника должна быть указана длина записей, равная ДЗС, ДЗК и ДЗН соответственно.

Описания массивов старого справочника, корректур и нового справочника составляются согласно требованиям СМО ЭВМ «Минск-32».

В программе предусмотрены возможности подключения нестандартных блоков пользователя (НБП) в следующих случаях:
— после открытия обрабатываемых массивов;
— перед закрытием нового справочника;
— после чтения очередной записи корректур и аналогично — старого справочника;
— перед очередным сравнением записей старого справочника и корректур;
— перед выводом очередной записи в новый справочник.

Следует заметить, что программа КОРЕК обрабатывает записи, длина и структура которых описана в обращении к программе корректировки. Однако допускается возможность обработки записей старого справочника и корректуры, имеющих структуру и длину, отличные от описанных в обращении. Это может быть достигнуто подключением программы пользователя после чтения очередной записи старого справочника и аналогично — корректуры, в которой предусматривается приведение этих записей к описанной в обращении структуре.

Если возникает необходимость получения записей нового справочника, отличных по структуре от записи, описанной в обращении к программе, то пользователь может это осуществить с помощью НБП перед выводом очередной записи в новый справочник.

Запись старого справочника и аналогично — корректуры, обрабатываемая в НБП после чтения очередной записи, находится в первом поле записи, т. е. начиная с адреса АНЗС1 и АНЗК1 соответственно. Записи старого справочника и корректуры, обрабатываемые в НБП перед их сравнением, находятся во вторых полях записей, т. е. начиная с адресов АНЗС2 и АНЗК2 соответственно. Запись нового справочника, обрабатываемая в НБП перед ее выводом, находится в поле записи нового справочника, т. е. начиная с адреса АНЗН.

Подключаемые НБП должны располагаться во внешней программе и иметь следующую структуру:

Этикетка	КОП	Адреса и замечания
АННБП	БАЗ НОП	0

Этикетка	КОП	Адреса и замечания
	РЗВ	2
	⎫ ⎪ ⎬ ⎪ ⎭	<Тело НБП>
	ВЫХ	АННБП; +0

где АННБП — адрес входа в НБП, указываемый в описании.

Следует заметить, что в режиме СУМК перед выводом очередной записи в новый справочник можно реализовать с помощью НБП одновременно подсуммирование одних неключевых реквизитов и замену других. При этом подсуммирование осуществляется непосредственно программой корректировки согласно описанию (режим СУМК и описания суммируемых реквизитов). Замену других неключевых реквизитов можно произвести с помощью НБП перед выводом очередной записи. Ему достаточно сравнить текущие записи старого справочника и корректуры по ключевым реквизитам, которые находятся в рабочих полях, начиная с адресов АНЗС2 и АНЗК2 соответственно. В случае их совпадения необходимо заменяемые реквизиты переслать из записи корректуры в запись нового справочника.

Пример. Пусть требуется произвести корректировку массива записей следующей структуры:

Идентификатор реквизита	Тип реквизита	Номер ячейки	Начальный разряд (символ)	Длина реквизита
ЦЕХ	ДЦ	0	1	16
ДЕТАЛЬ	ДЦ	1	1	36
НОРМА	ДЗ	2	1	56
ЦЕНА	ВБ	3	21	16
ЗАДЕЛ	ПЛ	4	—	—

Массив рассортирован по реквизитам:
ЦЕХ — по возрастанию:
ДЕТАЛЬ — по убыванию.

Массив корректур имеет ту же структуру и тот же порядок сортировки. Необходимо выполнить процедуру «подсуммирование» по реквизитам НОРМА, ЦЕНА и ЗАДЕЛ старого справочника и корректур.

Для выполнения этой работы внешняя программа должна содержать следующее обращение к программе КОРЕК:

Этикетка	КОП	Адреса и замечания
	ИП	КОРЕК; 1
	КА	0; ОКОР

где

Этикетка	КОП	Адреса и замечания
КОРЕК	ОПР	КОРЕК
	НОП	
ОКОР	КТ	СУМК
	КА	5; АОКР
	КА	АОСР; 0
	КА	АОМС; АТАС
	КА	АОМК; АТАК
	КА	АОМН; АТАН
	КА	0; 0
	КА	0; 0
	КА	0; 0
АОСР	КЧ	3
	КТ	ДЗ
	КЧ	2010070В
	КТ	ВБ
	КЧ	3250020В
	КТ	ПЛ
	КЧ	4000000В
АОМС	КТ	МАСС1
	НОП	
	КЧ	—0
	НОП	
	КНВУ	МЛ,1
	КЧ	5000000В
	НОП	
	НОП	
АОМК	КТ	МАСС2
	НОП	
	КЧ	—0
	НОП	
	КНВУ	МЛ,2
	КЧ	5000000В
	НОП	
АОМН	КТ	МАСС3
	НОП	
	КЧ	—0

Этикетка	КОП	Адреса и замечания
	НОП	
	КНВУ	МЛ,3
	КЧ	5000000В
	НОП	
АТАС	КА	ПОЛЕ1; ПОЛЕ1+502В
	КА	П1; П1+7
	КА	П2; П2+7
АТАК	КА	ПОЛЕ2; ПОЛЕ2+502В
	КА	П3; П3+7
	КА	П4; П4+7
АТАН	КА	ПОЛЕ3; ПОЛЕ3+502В
	КА	П5; П5+7
АОКР	КЧ	3
	КТ	ДЦВ
	КЧ	10020В
	КТ	ДЦУ
	КЧ	1010044В
ПОЛЕ1	РЗВ	503В
ПОЛЕ2	РЗВ	503В
ПОЛЕ3	РЗВ	503В
П1	РЗВ	8
П2	РЗВ	8
П3	РЗВ	8
П4	РЗВ	8
П5	РЗВ	8

3.7. ВЫВОД ЭКОНОМИЧЕСКОЙ ИНФОРМАЦИИ

ЭВМ «Минск-32» может работать в мультипрограммном режиме, при этом одновременно будет выполняться до четырех программ пользователя. Такой режим обеспечивается системой прерывания и совмещением операций ввода-вывода с обработкой информации процессором.

Рассматриваемая система вывода информации на алфавитно-цифровое печатающее устройство реализуется с помощью

двух программ *: «Подготовка информации для вывода на устройство печати (ПОДГ)» и «Вывод информации на устройство печати (ПЕЧАТ)». Первая из них осуществляет предварительную обработку входных массивов и запись массива, представляющего собой копию выходной формы, на магнитную ленту. Вторая выводит полученную информацию на устройство печати. При таком режиме вывода, когда при выводе практически никакой обработки информации в процессоре не делается, достаточно эффективно использование мультипрограммного режима. Другими словами, с выводом может быть совмещена любая обработка информации, не использующая устройство печати. Кроме того, упрощается организация работ по выводу информации за счет простоты повторения вывода как всего массива (например, для получения дополнительного экземпляра выходных форм), так и отдельных страниц (например, при сбоях устройства печати).

Программа подготовки информации для вывода на устройство печати (ПОДГ). Программа предназначена для предварительной обработки входной информации и записи ее на магнитную ленту. Вывод полученной информации осуществляется с помощью программы ПЕЧАТ.

Исходными данными для работы программы подготовки информации для вывода на устройство печати являются входной массив информации, описания массивов, таблицы информации, описание форм, строк, записей, реквизитов, титульных листов, заголовков, концовок, верхних и нижних шапок, подключаемые нестандартные блоки пользователя (НБП).

Входной массив должен состоять из записей фиксированной длины и располагаться на магнитной ленте или в ОП. Записи массива могут иметь различную структуру, в этом случае они должны снабжаться внутренним шифром.

Описания составляются пользователем во внешней (головной) программе, в ней должно быть предусмотрено закрепление НМЛ и выдача оператору указаний об установке магнитных лент.

Результатом работы программы является записанный на магнитную ленту массив информации, подготовленный для вывода на устройство алфавитно-цифровой печати. Массив состоит из записей фиксированной длины, равной ширине одного экземпляра, имеет специфическую структуру и может быть выведен только с помощью программы ПЕЧАТ.

Выходная форма может содержать: титульные листы, заголовки, концовки, верхние и нижние шапки, текст в строках, итоговые строки вместе с названием.

---

* В разработке этих программ принимали участие Н. А. Гольбурт, Л. Г. Фурса.

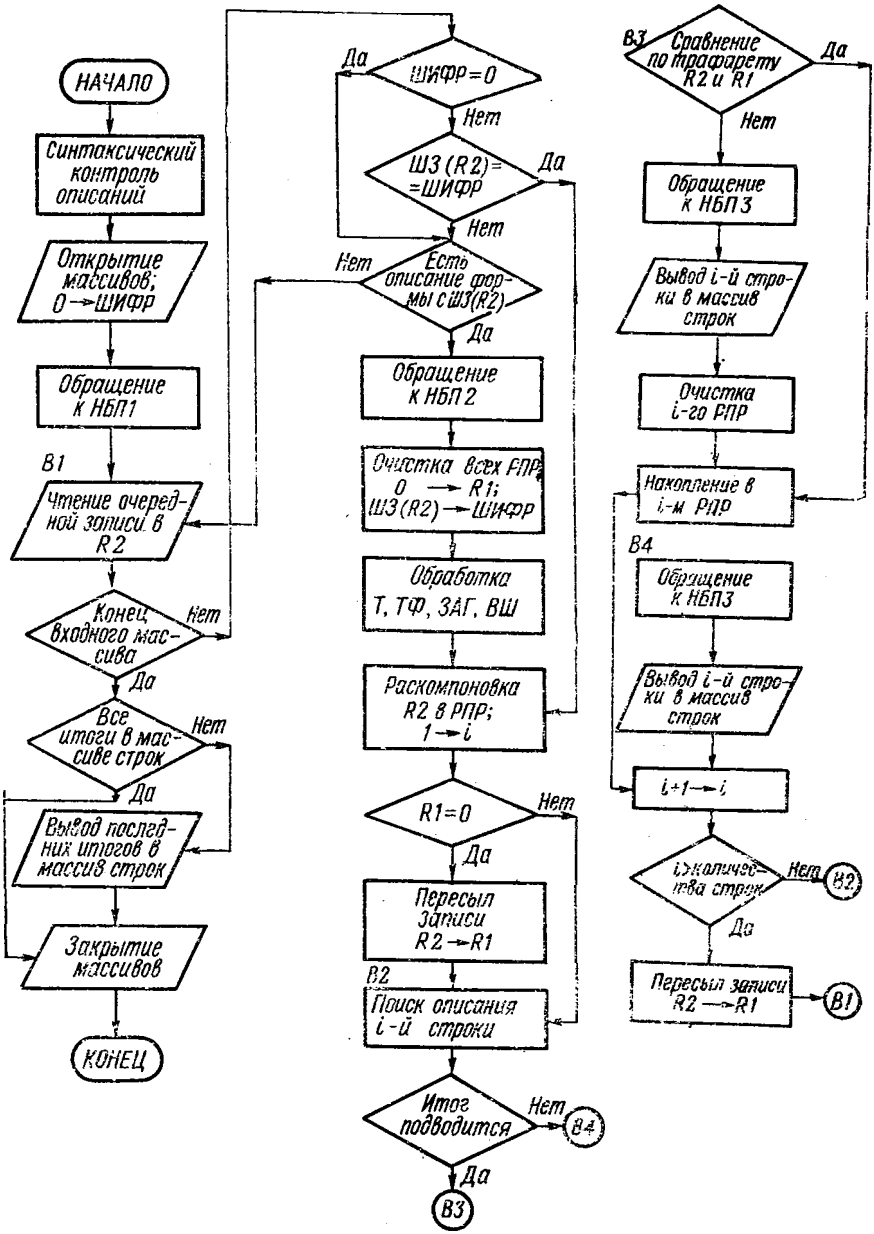


Рис. 3.14. Блок-схема программы ПОДГ

На каждом листе, кроме титульного, в правом верхнем углу выводится номер листа (нумерация с каждого титульного листа начинается с нуля). Блок-схема программы ПОДГ приведена на рис. 3.14.

На блок-схеме использованы следующие обозначения:

НБП1 — нестандартный блок пользователя в начале работы программы,

НБП2 — нестандартный блок пользователя перед началом вывода каждой формы в массив строк,

НБП3 — нестандартный блок пользователя перед выводом каждой строки в массив строк,

ШЗ — внутренний шифр записи,

Т — титульный лист,

ТФ — заголовок после титульного листа,

ЗАГ — заголовок,

ВШ — верхняя шапка,

РПР — рабочее поле реквизитов,

R1, R2 — первое и второе поле для размещения записей входного массива.

В начале работы программа производит синтаксический контроль описаний, составленных пользователем. В случае обнаружения ошибок выдаются соответствующие сообщения оператору.

Программа последовательно вызывает записи исходного массива и производит выборку только тех из описанных записей которые не удовлетворяют условию $R_i \gtrless A_i$, если это условие задано (см. ниже описание формы). Программа позволяет обрабатывать записи разной структуры, снабженные внутренним шифром (ШЗ). Если при этом встречаются записи, обработка которых не предусмотрена в описании, то выдается соответствующее сообщение.

Записи раскомпоновываются в первое рабочее поле реквизитов (РПР), и производится формирование выходных логических записей (называемых далее строками) и вывод их на магнитную ленту (массив строк). Если требуется выводить итоги, то ведется их накопление и вывод в выходной массив строк.

В программе предусмотрена возможность редактирования реквизитов, если это необходимо. Реквизиты в строках, в том числе и в итоговых, типа ПЛ округляются.

Обращение к программе имеет вид:

Этикетка	КОП	Адреса и замечания
	ИП	ПОДГ; 1
	КА	О; ТИ

где ТИ — символический адрес таблицы информации, которая имеет следующий вид:

Этикетка	КОП	Адреса и замечания
ТИ	КА	ДМЛ1; ДМЛ2
	КА	АН1; АК1
	КА	ЗН1; ЗК1
	КА	ЗН; ЗК
	КА	0; 0
	КА	АН2; АК2
	КА	ЗН2; ЗК2
	КА	0; 0
	КА	0; НБП1
	НОП	
	КЧ	<а><б><в><г><д>
	КА	<ШЗ1>; Ф1
	.	
	.	
	.	
	КА	<ШЗМ>; ФМ
	КТ	

где ДМЛ1 — адрес описания исходного массива;

ДМЛ2 — адрес описания массива строк;

АН1; АК1 — начальный и конечный адреса поля ввода для исходного массива.

При резервировании поля ввода для исходного массива в ячейке с символическим адресом АН1 указывается размер обмениваемой зоны плюс 3 контрольных ячейки, в ячейке с адресом АК1—1. Например, в случае обмена зонами по 500_8 ячеек:

Этикетка	КОП	Адреса и замечания
АН1	РЗВ	503В
АК1	РЗВ	1

ЗН1, ЗК1, ЗН, ЗК — начальные и конечные адреса полей обмена для записей исходного массива (резервируются два поля для записей). При резервировании полей обмена для записей в ячейке с начальным адресом поля обмена указывается размер обмениваемой записи плюс 2 контрольных ячейки, в ячейке с конечным адресом — 1 контрольная ячейка. Например, если длина записи исходного массива 8 ячеек:

Этикетка	КОП	Адреса и замечания
ЗН1	РЗВ	10
ЗК1	РЗВ	1

АН2; АК2 — начальный и конечный адреса поля вывода для массива строк. Размер поля вывода для массива строк обязательно должен быть кратным длине записи массива строк. Резервирование поля вывода для массива строк аналогично резервированию поля ввода для исходного массива;

ЗН2; ЗК2 — начальный и конечный адреса поля обмена для записей массива строк. Резервирование поля обмена для записей массива строк аналогично резервированию полей обмена для записей исходного массива;

НБП1 — адрес нестандартного блока пользователя (НБП), выполняемого в начале работы программы. В случае отсутствия НБП1 ячейка с указанием адреса НБП1 в таблице информации опускается;

а, б, в, г, д — параметры для раскомпоновки шифра записи ШЗ (в качестве шифра записи может служить любой реквизит-признак исходной записи, причем он может занимать в ячейке не более 15 двоичных разрядов):

а — восьмеричный номер ячейки реквизита в исходной записи (разряды 1÷9 — три восьмеричные цифры). Отсчет ячеек начинается с нуля;

б — восьмеричный номер левого бита (двоичного разряда) реквизита относительно начала ячейки в исходной записи (разряды 10÷15 — две восьмеричные цифры);

в — размер реквизита в битах (разряды 16÷21 — две восьмеричные цифры);

г — восьмеричный номер ячейки реквизита в рабочем поле реквизитов (разряды 22÷30 — три восьмеричные цифры), предназначенной для размещения значения реквизита, для шифра записи значение г=000;

д — восьмеричный номер левого бита реквизита, если он расположен в конце ячейки (разряды 31÷36 — две восьмеричные цифры), т. е. д=45—в. Для текстовых реквизитов д=43—в;

ШЗ1, ..., ШЗМ — различные внутренние шифры записей (восьмеричные числа — разряды 5÷20). Все различные шифры записей должны быть одинаково расположены в исходной записи и описываться одними и теми же параметрами а, б, в, г, д;

Ф1, ..., ФМ — начальные символические адреса описания форм, соответствующих этим различным шифрам записей;

КТ — здесь и в дальнейшем признак окончания описания.

Если шифр исходной записи один и тот же, то программой предусмотрен вывод только одной формы. В этом случае можно положить ШЗ=0 и а, б, в, г, д равными нулю.

Для работы программы необходимо составить два описания массивов: описание исходного массива и описание массива строк.

При составлении описания массива строк надо учитывать, что обмен будет производиться записями фиксированной длины, равной ширине экземпляра по 5 символов из ячейки.

Описания этих массивов составляются согласно требованию СМО ЭВМ «Минск-32».

Значение длины записи (л) для массива строк определяется по формуле

$$л = \left[\frac{\text{ширина экземпляра}}{5} + 0{,}5\right] + 1.$$

Ширина экземпляра должна быть не меньше 27 символов.

Для указания порядка печати строк составляется описание выводимой формы:

Этикетка	КОП	Адреса и замечания
Ф	КА	<ПР><С><ОБЩ> ЗАП
	КА	Т; ТФ
	КИ	0; <Н>
	КА	ЗАГ; КОН
	КА	ВШ; НШ
	КА	НРПР; <ДРПР>
	КТ	<НФ>
	КЧ	<а><б><в><г> <д>
	КЧ	<А>
	КА	0; НБП2
	НОП	
	КА	0; СТ1
	.	
	.	
	.	
	КА	0; СТ *L*
	НОП	
	КА	0; Р1
	.	
	.	
	.	
	КА	0; Р*N*
	КТ	

Программа предусматривает возможность при обработке массива пропускать записи согласно одному из следующих условий:

$$R \gtrless A,$$

где R — значение некоторого реквизита из записи, причем описание этого реквизита может не встречаться в описании реквизитов и в описании записей.

Реквизит R должен занимать не более одной ячейки и может быть любого типа, за исключением ПЛ. Типы A и R должны совпадать. Если тип реквизита A равен Т, то девятую строку описания формы КЧ<A> можно заменить на КТ<А>.

В первой ячейке описания формы:

ПР — признак пропуска записи (разряды 12÷14).

Если ПР=1, запись не участвует в обработке при выполнении условия $R>A$.

Если ПР = 2, то запись не участвует в обработке при $R<A$; если ПР = 4, то — при $R = A$; если ПР = 0, то не проверяется условие $R \gtrless A$;

а, б, в, г, д — параметры для раскомпоновки реквизита R, для которого проверяется условие $R \gtrless$ А(описываются аналогично, как и для раскомпоновки шифра записи).

Если условие $R \lessgtr$ А не проверяется, то в восьмой и девятой ячейках описания формы записывается НОП.

С — признак наличия итога по странице (разряды 15÷17). С=1 — подводится итог по странице; С=0 — итог не подводится.

ОБЩ — признак наличия общего итога формы (разряды 18÷20). ОБЩ=1 — подводится общий итог формы; ОБЩ=0 — не подводится;

ЗАП — начальный адрес описания записи исходного массива;

Т — начальный адрес описания титульного листа. При отсутствии титульного листа Т=0;

ТФ — начальный адрес заголовка после титульного листа, отличного от заголовка информационных строк. Этим заголовком может быть, например, название какой-то табуляграммы, которое необходимо вывести только на первом листе после титульного листа. При отсутствии заголовка после титульного листа ТФ=0;

Н — размер страницы (длина экземпляра);

ЗАГ, КОН, ВШ, НШ — начальные адреса описаний соответственно заголовка, концовки, верхней и нижней шапок. При оформлении страниц выводимой формы иногда возникает необходимость на первом листе напечатать полностью верхнюю и нижнюю шапки. Здесь они называются заголовком и концовкой (ЗАГ, КОН). На последующих листах печатать их в сокра-

щенном или измененном виде, например, выводить только номера граф. Эти шапки называются верхней и нижней шапками (ВШ, НШ). Если ВШ=0 и НШ=0, то все листы оформляются с заголовком и концовкой; если ВШ≠0, то ЗАГ≠0; если НШ≠0, то КОН≠0;

НРПР — начальный адрес всех рабочих полей реквизитов (РПР). Под этим символическим адресом резервируется место для всех РПР;

ДРПР — длина одного РПР;

НФ — номер или идентификатор формы (не более 5 символов). Если НФ меньше 5 символов, то ячейка дополняется пробелами. При отсутствии НФ в соответствующей ячейке описания ставится НОП;

НБП2 — адрес нестандартного блока пользователя, выполняемого перед началом вывода формы. В случае отсутствия НБП2 ячейка с адресом нестандартного блока опускается;

СТ1, ..., СТ*L* — начальные адреса описаний строк в той последовательности, в которой строки выводятся на печать;

Р1, ..., Р*N* — начальные адреса описаний реквизитов, выводимых в итоге по странице. Порядок следования безразличен. Если итог не подводится, ячейки с адресами описаний реквизитов отсутствуют в описании формы.

Описания титульного листа (Т), заголовка (ЗАГ), концовки (КОН), верхней (ВШ), нижней шапок (НШ) и заголовка после титульного листа (ТФ) составляются следующим образом:

Этикетка	КОП	Адреса и замечания
ЭТ	КЧ	<ИНТ>
	КА	<НАЧИ>; НАЯ
	КИ	<Л><Л1><ЧИ>; <ЧВС>
	.	. . .
	.	. . .
	.	. . .
	КА	<НАЧИ>; НАЯ
	КИ	<Л><Л1><ЧИ>; <ЧВС>
	КТ	

где ЭТ — одна из этикеток: Т, ЗАГ, КОН, ВШ, НШ, ТФ;

ИНТ — число интервалов до первой строки;

НАЯ — начальный адрес выводимой информации;

НАЧИ — номер позиции в строке, начиная с которой помещается эта информация;

ЧВС — число выводимых символов (восьмеричное число);

Л — признак печати символа, расположенного в первых семи разрядах ячейки НАЯ, ЧВС раз (разряды 6÷8);

Л = {1 — занесение символа из ячейки НАЯ ЧВС раз;
0 — занесение текста из ячейки НАЯ;

$Л_1$ — признак окончания строки (разряды 9÷11);

$Л_1$ = {1 — строка закончена;
0 — строка не закончена;

ЧИ — число интервалов после строки с текстом при распечатке массива строк (разряды 12÷20 — три восьмеричные цифры); если строка не закончена (Л1=0), то значение ЧИ безразлично.

Описание записи исходного массива должно составляться следующим образом:

Этикетка	КОП	Адреса и замечания
ЗАП	КА	0; P1
	.	
	.	
	.	
	КА	0; P*N*
	КТ	

где P1, ..., P*N* — начальные адреса описаний реквизитов исходной записи.

Если один реквизит занимает в исходной записи несколько ячеек, то занимаемые им разряды в каждой ячейке описываются как отдельные реквизиты, для которых резервируются отдельные ячейки рабочего поля реквизитов.

Каждая форма состоит из некоторого количества строк. Под строкой понимается любая строка, подготавливаемая к печати, включая все итоги (за исключением итога по странице). Строка состоит из реквизитов. Для каждой строки должно быть отведено свое рабочее поле реквизитов (РПР). Итог по странице не описывается как строка, однако для реквизитов, выводимых в итоге по странице, выделяется одно РПР. Все РПР имеют одинаковую длину и структуру, которые зависят только от длины и структуры исходной записи. Засыл значений реквизитов в первое РПР происходит в результате раскомпоновки исходной записи, а также в результате получения значений некоторых реквизитов по программам пользователя. Все РПР располагаются сплошным массивом в порядке их следования друг за другом.

Программа работает таким образом, что после подготовки каждой строки происходит накопление значений необходимых реквизитов в следующем РПР, после чего предыдущее поле очищается.

Необходимость вывода тех или иных итогов определяется с помощью специально подобранных трафаретов, включаемых в описание каждой строки. Длина трафарета равна количеству сравниваемых ячеек записи. Причем трафарет всегда заготавливается, начиная с первой и кончая той ячейкой записи, в которой содержатся последние сравниваемые реквизиты. В двоичных разрядах трафарета, соответствующих тем реквизитам, по которым производится сравнение, ставятся единицы, а в двоичных разрядах, соответствующих реквизитам, по которым не производится сравнение, ставится нуль. Формирование и вывод строк осуществляется по принципу вложенных трафаретов. Принцип работы с трафаретами показан на блок-схеме, приведенной на рис. 3.15. На этой блок-схеме КС — количество различных строк в выводимой форме без учета строки с итогом по странице.

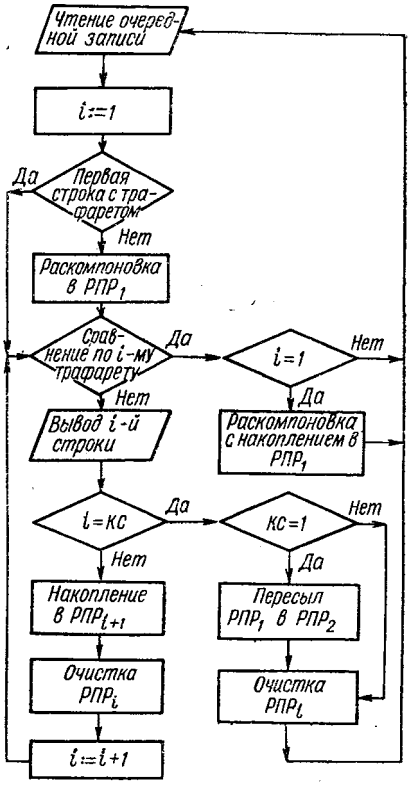


Рис. 3.15. Принцип работы с трафаретами

В описании к программе необходимо зарезервировать столько РПР, сколько составлено описаний строк, а также одно (последнее) РПР для итога по странице (если итог подводится). Исключение представляет случай вывода на печать однотипных строк (составляется описание только одной строки). В этом случае резервируются два РПР. Следует иметь в виду, что накопление и пересыл в $(i+1)$-е РПР происходит только тех реквизитов, которые участвуют в выводе i-й строки.

Описание строки имеет следующий вид:

Этикетка	КОП	Адреса и замечания
СТ	КА КЧ КА КИ	<ДТ>; НАРЯТ >КСТ><СТР><ПТ> <П><ДЗ> НАЯ; <ЧВС> 0; <НАЧИ>

Продолжение

Этикетка	КОП	Адреса и замечания
	КЧ . . . КЧ	<трафарет>
	КА	0; НБПЗ
	НОП	
	КА	0; Р1
	.	
	.	
	КА	0; Р*N*
	КТ	

где ДТ — количество ячеек (длина), занимаемых трафаретом. При отсутствии трафарета ДТ=0 и ячейки КЧ для трафарета пропускаются;

НАРЯТ — начальный адрес рабочих ячеек, предназначенных для текста. Эти ячейки должны быть заполнены по определенному правилу. В рабочую ячейку с этикеткой НАРЯТ помещается восьмеричное число символов в этом тексте. Начиная со второй ячейки, помещается текст в символьном виде. Если рабочие ячейки с текстом отсутствуют, то НАРЯТ=0.

НАЯ, ЧВС, НАЧИ, ДЗ приводятся в том случае, если в строке необходимо отпечатать ее словесное название. Например: «итого», «итого по цеху», «по участку» и т. п.

НАЯ — начальный адрес ячеек с текстом;

ЧВС — число выводимых символов в ячейках с текстом;

НАЧИ — номер позиции в строке, начиная с которой располагается этот текст;

ДЗ — длина занесения текста (разряды 28÷36 второй ячейки описания строки — три восьмеричные цифры). В случае отсутствия названия строки НАЯ=0, ЧВС=0, НАЧИ=0, ДЗ=0;

П — признак занесения строки в массив строк (разряды 25÷27):

$$П = \begin{cases} 0 — \text{строка заносится;} \\ 1 — \text{строка не заносится.} \end{cases}$$

При обработке очередной строки значение признака П устанавливается программой ПОДГ в нуль. Его можно сделать рав-

ным 1 с помощью НБП, обращение к которому делается в описании строки;

ПТ — признак печати титульного листа после строки (разряды 22÷24):

$$ПТ=\begin{cases}0 — \text{после данной строки титульный лист не печатается;}\\ 1 — \text{печатается;}\end{cases}$$

СТР — признак открытия новой страницы после данной строки (разряды 19÷21):

$$СТР=\begin{cases}0 — \text{после данной строки новая страница не открывается;}\\ 1 — \text{открывается;}\end{cases}$$

если ПТ=1, то значение СТР безразлично, так как после титульного листа всегда открывается новая страница;

КСТ — количество протяжек бумаги после строки (разряды 13÷18 — две восьмеричные цифры);

НБП3 — адрес нестандартного блока пользователя, выполняемого перед выводом строки. Все реквизиты, вычисляемые с помощью НБП должны быть помещены на своих местах в соответствующем РПР;

Р1, ..., РN — начальные адреса описаний реквизитов, выводимых в строке. Порядок следования безразличен.

В некоторых случаях выводимая форма может состоять только из однотипных строк (составляется описание только одной строки). Однако после вывода некоторого количества таких строк необходимо сделать определенное количество переводов строк, открыть новый титульный лист или новую страницу. Для этого достаточно искусственно ввести описание итоговой строки, не выводимой на печать, указав в этом описании трафарет для сравнения реквизитов и нужный признак (признак печати титульного листа ПТ=1 или открытия новой страницы СТР=1 после строки по результатам сравнения). Ячейки с адресами описаний реквизитов в описании этой строки пропускаются.

В описании реквизитов перечисляются реквизиты из исходных записей, которые должны раскомпоновываться в РПР. Кроме того, здесь могут быть описаны реквизиты, получаемые в нестандартных блоках пользователя. Отметим, что все реквизиты, за исключением некоторых текстов (см. ниже), участвующие в выводимых строках, должны быть описаны. Реквизиты в исходной записи могут быть следующих типов:

Т — текстовый;
ВБ — восьмеричный целый без знака;
ВФ — восьмеричный с фиксированной запятой;
ДЦ — десятичный целый;
ДФ — десятичный с фиксированной запятой;
ПЛ — двоичный с плавающей запятой.

Описание реквизитов имеет следующий вид:

Этикетка	КОП	Адреса и замечания
Р	КЧ	<а><б><в><г><д>
	КЧ	±<КЗ><ДЗ><П1><П2><П3><АБ>
	КА	<ЫЗ>; <ПП>

В первой ячейке описания а, б, в, г, д — параметры для раскомпоновки реквизита из исходной записи в РПР (описываются аналогично, как и для раскомпоновки шифра записи). Нумерация ячеек в исходной записи и в РПР начинается с нуля. Если один реквизит в исходной записи занимает несколько ячеек, то занимаемые им разряды в каждой ячейке описываются как отдельные реквизиты.

АБ — восьмеричный номер позиции в строке, начиная с которой располагается реквизит (разряды 28÷36 — три восьмеричные цифры);

П1 — признак обработки (накопления) реквизита (разряды 19÷21) : П1=0 — реквизит не накапливается для следующего итога и печатается всегда; П1=1 — реквизит накапливается для следующего итога и печатается всегда; П1=4 — реквизит не накапливается для следующего итога и печатается по изменению значения. На каждой новой странице и после печати какого-нибудь итога печатаются все реквизиты, перечисленные в описании строки (независимо от того, П1=4 или нет).

П2 — признак представления реквизита в РПР (разряды 22÷24) : П2=0 — реквизит необходимо накопить в двоичной системе счисления в целых числах (тип ВБ, ВФ); П2=1 — в двоичной системе счисления с плавающей запятой (тип ПЛ); П2=4— в десятичной системе счисления (тип ДЦ, ДФ). Если П1=0 или 4, то значение П2 безразлично.

П3 — вид занесения реквизита в строку (разряды 25÷27): П3=0 — реквизит находится в РПР в символьном виде (тип Т); П3=1 — в десятичной системе счисления (тип ДЦ); П3=2 — реквизит находится в рабочих ячейках, этикетка начального адреса которых НАРЯТ указана в описании строки (тип Т);

ДЗ — длина занесения реквизита в строку (разряды 10÷18 — три восьмеричные цифры).

Для вывода реквизит (накопленный реквизит) должен быть представлен как десятичное целое (в двоично-десятичном коде) или в символьном виде (код ГОСТ 10859—64). Если реквизит не представлен в одной из этих двух систем, то его необходимо перевести с помощью программы перевода, признак которой ПП указывается в третьей ячейке описания реквизита. При переводе дробного реквизита программа перевода не только отделяет кодом запятой (0001110) целую часть от дробной, но и уничтожает нули до первой значащей цифры.

КЗ — масштаб реквизита, переведенного в десятичную систему счисления (разряды 4÷9 — две восьмеричные цифры). Если КЗ>40, то значение реквизита умножается на $10^{кз-40}$, если КЗ<40, то на $10^{-кз}$. Наличие минуса перед КЗ является признаком вывода реквизита без гашения незначащих нулей.

ЫЗ — количество знаков после запятой в выводимом реквизите.

Реквизит (накопленый реквизит) для перевода должен занимать в РПР не более одной ячейки.

ПП=0 — не нужен перевод реквизита из одной системы счисления в другую; ПП=1 — реквизит в РПР представлен в двоичной системе счисления с плавающей запятой (тип ПЛ). Предусмотрен перевод в десятичную систему счисления с вставлением запятой (код ГОСТ 10859—64). В этом случае в описании реквизита КЗ=00. ПП=2 — реквизит в РПР представлен как восьмеричное целое (тип ВБ). Предусмотрен перевод в десятичную систему счисления в целых числах с вставлением запятой. В режиме вставления запятой старшая цифра (она может быть незначащей) реквизита теряется. Необходимое значение КЗ указывается пользователем. Нули до первой значащей цифры заменяются пробелами. Максимально переводимое число равно $7346544777_8=999999999$, а в режиме вставления запятой — $575360377_8=999999999$. В случае, если значение реквизита превышает максимально допустимое, то этот реквизит при выводе на печать не выводится, а заменяется кодом «точка» (0001110). ПП=3 — реквизит в РПР представлен как восьмеричный с фиксированной запятой (тип ВФ). Предусмотрен перевод в десятичную систему счисления в целых числах с вставлением запятой. В режиме вставления запятой старшая цифра (она может быть незначащей) реквизита теряется. Необходимое значение КЗ указывается пользователем. ПП=4 — реквизит в РПР в десятичной системе счисления с фиксированной запятой (тип ДФ). Предусматривается перевод в десятичную систему счисления в целых числах с вставлением запятой. В режиме вставления запятой старшая цифра реквизита (она может быть незначащей) теряется. Необходимое значение КЗ указывается пользователем.

Если в описании реквизита ПП≠0, то значение ПЗ для этого реквизита безразлично.

Если в описании реквизита ПЗ=2, то рабочие ячейки с текстом, этикетка начального адреса которых НАРЯТ, могут выводится по какому-либо реквизиту-признаку исходной записи. Это осуществляется в программе поиска наименования, которая включается в описание соответствующей строки в качестве НБПЗ. Если сам реквизит-признак не выводится на печать, а выводятся только рабочие ячейки с текстом, то в описании реквизита КЗ=0, П1=0, П2=0, ПЗ=2, ПП=0 остальные значения указываются согласно описанию реквизита. В этом случае

текст из рабочих ячеек с этикеткой начального адреса НАРЯТ не описывается как отдельный реквизит. Если рабочие ячейки с текстом выводятся всегда (не по реквизиту-признаку или вместе с ним), то предназначенный для вывода текст описывается как реквизит, ячейки в РПР для него не резервируются.

Описание такого реквизита имеет вид:

Этикетка	КОП	Адреса и замечания
РТ	КЧ	0
	КЧ	<ДЗ><П1><П2><П3><АБ>
	КЧ	

где П1=0; П2=0; П3=2; ДЗ — длина занесения текста в одной строке (разряды 10÷18 — три восьмеричные цифры). Количество строк, занимаемых текстом, вычисляется по формуле

$$\left[\frac{\text{ДЛ}}{\text{ДЗ}} + 1\right],$$

где ДЛ — длина текста в рабочих ячейках. Текст в рабочих ячейках с этикеткой начального адреса НАРЯТ можно изменить с помощью НБП3, адрес которого указан в описании строки.

При обработке реквизитов, раскомпонованных в РПР, принят следующий порядок:

— накопление реквизитов;
— работа НБП3, адрес которого указан в описании строки;
— перевод реквизитов из одной системы счисления в другую (при этом для реквизитов типа ПЛ осуществляется округление);
— занесение реквизитов в массив строк.

Для выполнения нестандартных действий в программе предусмотрено подключение нестандартных блоков пользователя в следующих местах:

а) в начале работы программы (адрес НБП1 указывается в таблице информации);

б) перед началом вывода каждой формы в массив строк на магнитную ленту (адрес НБП2 указывается в описании соответствующей формы);

в) перед выводом каждой строки в массив строк на магнитную ленту (адрес НБП3 указывается в описании соответствующей строки).

Для выполнения программы ПОДГ и ПЕЧАТ пользователь должен составить головную (внешнюю) программу, согласно требованиям СМО ЭВМ «Минск-32». Все НБП должны быть приведены в этой головной программе и оформлены в виде самостоятельных подпрограмм.

Отметим, что первым оператором после оператора БАЗ и НБП обязательно должен быть оператор НОП. НБП могут ис-

пользовать рабочие и общие области головной программы и не должны иметь своих рабочих и общих областей.

В НБП, выполняемых перед выводом каждой формы и каждой строки, необходимо учитывать, что реквизиты исходной записи раскомпонованы в первое РПР и накоплены в соответствующих РПР. Поэтому адреса реквизитов определяются по формуле

$$РН = НРПР + К \times ДРПР + Г,$$

где РН — адрес реквизита в РПР; НРПР — начальный адрес всех РПР; ДРПР — длина одного РПР; К — порядковый номер (начиная с 0) выводимой строки; Г — порядковый номер (начиная с 0) ячейки реквизита в РПР (задается в описании соответствующего реквизита).

При работе НБП, подключаемых перед выводом каждой строки в массив строк, необходимо учитывать порядок формирования строки программой ПОДГ. При обработке реквизитов, раскомпонованных в РПР, для каждой строки принят следующий порядок:

— накопление в РПР обрабатываемой строки реквизитов, адреса которых указаны в описании строки;
— очистка пробелами поля обмена для записей массива строк (адреса ЗН2, ЗК2);
— работа НБП3, адрес которого указан в описании строки;
— проверка признака П занесения строки в массив строк. Если П = 1, то значение признака устанавливается в нуль и на этом обработка строки заканчивается;
— занесение наименования строки (см. параметры НАЯ, ЧВС, НАЧИ, ДЗ в описании строки) в поле обмена (адреса ЗН2; ЗК2). Количество печатных строк, занимаемых текстом, вычисляется по формуле: $\frac{ЧВС}{ДЗ}$ — если делится нацело, $\left|\frac{ЧВС}{ДЗ}\right| + 1$ — если не делится. Реквизиты выводятся в последней печатной строке с наименованием;
— перевод с редактированием реквизитов в код ГОСТ 20859—64;
— занесение реквизитов в поле обмена (адреса ЗН2; ЗК2) в порядке их следования в описании строки;
— вывод строки в массив строк.

Следует заметить, что к моменту подключения НБП1 в начале работы программы ПОДГ произведено лишь открытие входного массива. Пользователь может предусмотреть в своей программе чтение и анализ дополнительных контрольных блоков (если они есть), пропустить некоторое количество записей входного массива, сделав необходимые изменения в описании входного массива. В этом случае анализ выполняется один раз при однократном выполнении программы ПОДГ.

Нестандартные блоки пользователя, выполняющиеся перед началом обработки каждой формы, подключаются в тот момент,

когда в первое поле для записи исходного массива (адреса ЗН1, ЗК1) считана запись с нужным шифром. Пользователь в своей программе может выбрать необходимую информацию из записи, преобразовать ее, если надо, в символьную и занести на соответствующее ей место в постоянной информации для титульного листа, заголовка, концовки, верхней и нижней шапок. Очередная запись для обработки считывается во второе поле (адреса ЗН, ЗК). Поэтому для выполнения аналогичной задачи с помощью НБП3, выполняемыми перед выводом каждой строки, информация записи выбирается из поля с этикетками ЗН, ЗК.

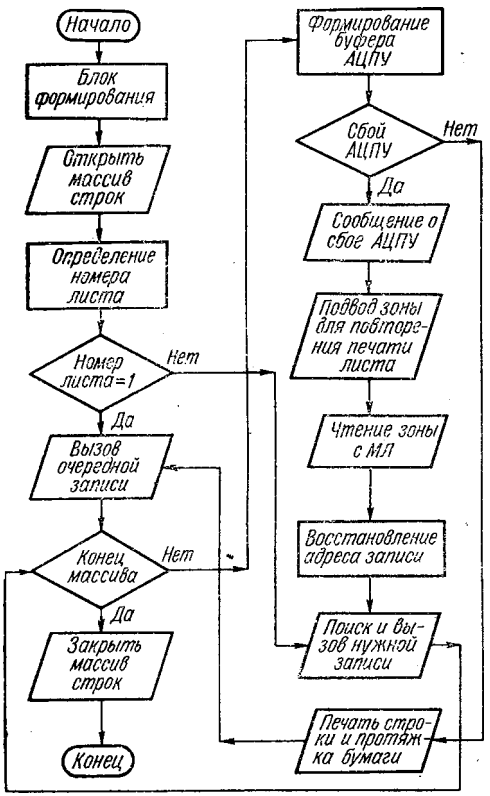


Рис. 3.16. Блок-схема программы ПЕЧАТ

Для удобства работы с НБП целесообразно поместить все рабочие поля и константы в общей области головной программы. Тогда в НБП можно непосредственно использовать этикетки общей области в качестве адресов операндов на ЯСК.

Если все реквизиты входной записи не помещаются в одну печатную строку, то их можно разместить в две строки (для каждой строки составляется описание и резервируется РПР). Но в НБП, подключаемом перед выводом первой строки, надо переслать $РПР_1$ в $РПР_2$.

Программа вывода информации на устройство печати (ПЕЧАТ). Программа предназначена для распечатки массива информации (массива строк), подготовленного для вывода на устройство алфавитно-цифровой печати и записанного на магнитную ленту программой ПОДГ. Программа ПЕЧАТ обеспечивает:

— вывод на устройство печати массива записей фиксированной длины, равной ширине одного экземпляра, подготовленного программой ПОДГ на магнитной ленте;
— параллельный вывод на печать нескольких экземпляров;
— повторение печати листа в случае сбоя устройства печати;

— повторение печати с любого номера листа;
— печать черты, отделяющей одну страницу от другой;
— оформление конца печати.

Блок-схема программы приведена на рис. 3.16.

При параллельном выводе нескольких экземпляров они отделяются друг от друга вертикальной чертой.

При распечатке массива строк ведется двойная нумерация листов: сквозная и внутренняя. Сквозной (порядковый) номер листа с учетом титульных листов печатается в левом верхнем углу на разделительной черте, начиная с единицы. Внутренняя нумерация начинается с единицы после каждого титульного листа. Внутренний номер листа печатается в правом верхнем углу под разделительной чертой на всех листах, кроме титульных. При распечатке массива строк с указанного листа номер его должен задаваться согласно сквозной нумерации.

Исходными данными для работы программы являются:
— массив строк, полученный на магнитной ленте в результате работы программы ПОДГ;
— описание массива строк (это описание в головной программе может быть составлено один раз для обеих программ);
— описание выводимого на печать массива.

Обращение к программе имеет вид:

ИП ПЕЧАТ; 1
КА∠ПОВТ> <НЛ>; ТИП

где ТИП — адрес таблицы информации, которая имеет следующий вид:

Этикетка	КОП	Адреса и замечания
ТИП	КА	ДМЛ2; ДПЧ
	КА	АН2; АК2

где ДМЛ2 — адрес описания массива строк;

ДПЧ — адрес описания выходного массива для работы с устройством печати;

АН2; АК2 — начальный и конечный адреса поля вывода для массива строк.

ПОВТ — признак необходимости повторения печати листа в случае сбоя АЦПУ (разряды 15÷17); ПОВТ=0, если требуется повторение печати листа после сбоя АЦПУ. ПОВТ=1 в противном случае. Признак ПОВТ влияет на скорость вывода. Так, при ПОВТ=1 скорость печати составляет около 400 строк в минуту, при ПОВТ=0 — около 330 строк в минуту.

НЛ — признак необходимости автоматического запроса номера печатаемого листа (разряды 18÷20). НЛ=1, если печать

всегда надо начинать с листа с номером 1. НЛ=0, если программа будет запрашивать через ПМ у оператора номер печатаемого листа.

Описание массива строк с этикеткой ДМЛ2 составляется как описание выходного массива для работы программы ПОДГ. Там же резервируется поле обмена для массива строк с этикетками АН2, АК2. Остается составить описание выходного массива для работы программы ПЕЧАТ. Это описание имеет следующий вид:

Этикетка	КОП	Адреса и замечания
ДПЧ	КТ	<имя массива>
	НОП	
	НОП	
	НОП	
	КНВУ	ПЧ, К
	КЧ	<ЧЭ><ШЭ>
	НОП	
	НОП	

В первой ячейке описания содержится текстовая информация — имя обмениваемого массива.

В пятой ячейке содержится номер внешнего устройства: тип ПЧ — устройство алфавитно-цифровой печати, К — номер в типе (К=1, 2, ...).

В шестой ячейке указываются:

ЧЭ — число экземпляров, выводимых на печать (разряды 19÷27 — три восьмеричные цифры);

ШЭ — ширина экземпляра (разряды 28÷36 — три восьмеричные цифры). Ширина экземпляра должна быть не менее 27 символов.

Вторая, третья, четвертая, седьмая и восьмая ячейки описания массива используются программой как рабочие.

В четвертой ячейке описания массива хранится десятичный номер очередного печатного листа.

Пример использования программ ПОДГ и ПЕЧАТ приведен в разд. 3.9.

3.8. УПРАВЛЕНИЕ ПАКЕТОМ ЗАДАЧИ

В зависимости от составленного пользователем описания пакета задачи управление вычислительным процессом состоит в следующем:

— вызов из библиотеки пользователя требуемой программы;

— реализация очередного шага решения задачи, т. е. пуск программы;

— переход к выполнению того или иного этапа решения задачи (указанного пользователем) в зависимости от анализа результатов предыдущего шага;
— остановка вычислительного процесса по задаче в указанном пользователем месте;
— выдача на пишущую машинку сообщений о ходе вычислительного процесса, а также управляющих сообщений оператору (снять или установить необходимые носители информации, включить внешние устройства и т. д.).

Управление вычислительным процессом реализуется с помощью двух модулей: * программы подготовки данных (КОНТР) и программы управления вычислительным процессом (ДИСП).

Программа подготовки данных (КОНТР). Программа является вспомогательной для программы управления вычислительным процессом. Она предназначена для обработки задания на управление вычислительным процессом, т. е. приведения его к виду, приемлемому для работы основной программы ДИСП, и записи задания на МЛ. Задание вводится с перфокарт и подвергается синтаксическому контролю.

Рассмотрим структуру операторов задания.

Управляющий оператор МЛЗ служит для указания МЛ, на которую должно быть записано обработанное задание.

Этикетка	КОП	Адреса и замечания
—	МЛЗ	< A >

где A — имя МЛ, на которую нужно записать обработанное задание. Оператор МЛЗ может отсутствовать. Если он есть в задании, то проверяется начало катушки МЛ для записи задания. Если его нет, проверки начала катушки нет. В этом случае обработанное задание будет записано на МЛ, установленную после соответствующего указания. Оператор МЛЗ следует записывать в первой строке задания. Оператор МЛЗ не может иметь этикетку.

Управляющий оператор ЗАДАЧ служит для указания имени задачи:

Этикетка	КОП	Адреса и замечания
—	ЗАДАЧ	< B >

где B — имя задачи. Оператор должен присутствовать в задании. Имя задачи, записанное в графе адреса и замечания, присваива-

* В разработке программы принимала участие Н. В. Лысикова.

ется обработанному заданию, записываемому на МЛ. Оператору ЗАДАЧ может предшествовать лишь оператор МЛЗ. Оператор ЗАДАЧ не может иметь этикетку.

Управляющий оператор ШАГ служит для указания имени выполняемой программы. При этом предполагается, что задача состоит из некоторой совокупности программ.

Этикетка	КОП	Адреса и замечания
[<имя>]	ШАГ	<С>

где С — имя программы. Оператор может иметь этикетку. Оператор ЫПМ служит для описания текста для вывода на ПМ.

Этикетка	КОП	Адреса и замечания
[<имя>]	ЫПМ	<текст сообщения>

Текст записывается в графе адреса и замечания. Если текст не умещается в одну строку, то его следует перенести в следующую, начиная с 22-й позиции. При этом во второй строке название оператора можно не писать. Оператор ЫПМ может иметь этикетку.

Управляющий оператор ОЫПМ:

Этикетка	КОП	Адреса и замечания
[<имя>]	ОЫПМ	<текст сообщения>

Все сказанное об операторе ЫПМ верно для оператора ОЫПМ. Кроме того, по оператору ОЫПМ происходит распечатка текста на ПМ и останов вычислительного процесса.

Для описания ветвления в вычислительном процессе предназначены управляющие операторы ПИ и ПТ.

При организации ветвления используется связь программы управления вычислительным процессом с программами пользователя. Программа управления вычислительным процессом резервирует поле СТ на общем базисе. Длина поля СТ равна одной ячейке с этикеткой СТ1.

Управляющий оператор перехода по имени ПИ:

Этикетка	КОП	Адреса и замечания
—	ПИ	

Оператором ПИ осуществляется переход на выполнение оператора, этикетка которого совпадает со значением ячейки СТ1 поля СТ. Если разветвление в вычислительном процессе описывается в задании оператором ПИ, то программа, предполагающая это разветвление, должна содержать резервирование поля СТ на общем базисе и посылку в ячейку СТ1 поля СТ этикетки, на которую следует перейти. Оператор ПИ не может иметь этикетку. Графа адреса и замечания для оператора ПИ не заполняется.

Управляющие операторы ПТ, ЗНАЧ:

Этикетка	КОП	Адреса и замечания
	ПТ	
	ЗНАЧ	<Д>; <Н>

Оператором ПТ организуется переход на ту или иную этикетку в зависимости от значения величины в ячейке СТ1 поля СТ. В операторе ЗНАЧ Д задает некоторые значения величины, а Н — этикетку, к которой нужно перейти, если содержимое ячейки СТ1 поля СТ равно Д.

Оператор ЗНАЧ употребляется только вслед за оператором ПТ. Один оператор ЗНАЧ задает одно значение величины. Поэтому вслед за оператором ПТ следует записать столько операторов ЗНАЧ, сколько всевозможных разветвлений предполагает программа, использующая этот оператор ПТ.

Операторы ПТ и ЗНАЧ не могут иметь этикетку. Название оператора ЗНАЧ нужно обязательно записывать в первой за оператором ПТ строке и можно опустить в последующих.

Оператор КОНЕЦ служит признаком конца вычислительного процесса:

Этикетка	КОП	Адреса и замечания
	КОНЕЦ	

Этикетки должны иметь те операторы, на которые возможны переходы по операторам ПИ и ПТ. Программа управления вычислительным процессом позволяет начать вычислительный процесс с любого оператора, имеющего этикетку. Это следует учитывать при составлении задания.

Задание должно быть отперфорировано на перфокартах по правилам перфорации программ на перфокарты. Идентификатор задания произвольный. Если информация управляющих операторов ШАГ, ЗАДАЧ, а также символические наименования в опе-

раторе ЗНАЧ содержат больше пяти символов, то при обработке задания будут учитываться только первые пять.

Результатом работы программы является задание, приведенное к виду, приемлемому для работы программы управления вычислительным процессом. При этом порядок следования операторов не меняется. Оператор МЛЗ не включается в задание, он используется только программой КОНТР.

Программа КОНТР контролирует правильность написания задания. Если в процессе обработки задания ошибок не найдено, то обработанное задание выводится на МЛ, в противном случае вывода задания на МЛ не происходит. На АЦПУ распечатывается задание в том виде, в котором оно записано на бланках ССК. Справа на листе печатаются сообщения об ошибках.

Программа оформляется как отдельная (внешняя) программа. Запуск программы в работу производится обращением к программе КООРДИНАТОР:

ВЫ$_3$— АААААА; ПМ◇*АААААА*КОНТР00100ЛС00501◇

Программа управления вычислительным процессом (ДИСП). Программа предназначена для управления вычислительным процессом в соответствии с заданием.

Программа вызывает с МЛ задание и выполняет его. По оператору ШАГ происходит вызов из библиотеки пользователя программы с именем, записанным в операторе, и запуск программы пользователя. По оператору ПИ организуется переход на выполнение оператора, имеющего метку, определяемую содержимым ячейки СТ1 поля СТ. По оператору ПТ содержимое ячейки СТ1 поля СТ сравнивается со значениями Д, указанными в операторах ЗНАЧ, которые следуют за этим оператором ПТ. Далее осуществляется переход на выполнение оператора с именем <Н>, указанным в операторе ЗНАЧ, значение Д которого совпало с содержимым ячейки СТ1 поля СТ.

Программа позволяет начать работу с любого оператора при условии, что этот оператор имеет этикетку. Есть возможность повторить участок вычислительного процесса с некоторой программы. Для этого оператор, описывающий программу, с которой при необходимости можно было бы повторить вычисление, должен иметь этикетку.

Останов вычислительного процесса происходит перед выполнением управляющего оператора, имеющего этикетку, и перед оператором КОНЕЦ. Это позволяет управлять ходом вычислительного процесса с ПМ.

Исходными данными являются:

— задание на реализацию вычислительного процесса, записанное на МЛ;

— библиотека программ пользователя.

Библиотека программ пользователя должна быть записана на МЛ. Рекомендуется записывать программы в собранном виде

на ту МЛ, на которой записано задание. В случае, если программа не собрана, она должна освобождать память всех своих внутренних сегментов.

Программа оформлена как отдельная (внешняя). Запуск программы в работу производится обращением к программе КООРДИНАТОР:

ВЫ — АААА; ПМ◇*АААА*ДИСПХХХVVЛС00501◇

где XXXVV (часы и минуты) — время работы программы.

Пример. Дана задача, решение которой реализуется с помощью программ ПРИМ1, ПРИМ2, ПРИМ3. Причем вычислительный процесс имеет следующую структуру: сначала выполняется программа ПРИМ1, затем ПРИМ2, в зависимости от результата работы программы ПРИМ2 должны выполняться программы ПРИМ1 или ПРИМ3. После программы ПРИМ3 следует конец вычислительного процесса. Программа ПРИМ2 последовательно обрабатывает несколько массивов с перфокарт. Сообщение об установке очередного массива на УВК печатать на ПМ. Задание на реализацию данного вычислительного процесса может быть записано в виде:

Этикетка	КОП	Адреса и замечания
	ЗАДАЧ	ВЫЧПР
ВЕСНА	ШАГ	ПРИМ1
	ОЫМП	УСТАНОВИТЬ ОЧЕРЕДНОЙ МАССИВ НА УВК
	ШАГ	ПРИМ2
	НТ	
	ЗНАЧ	А; ВЕСНА
		В; ВЕГА
ВЕГА	ШАГ	ПРИМ3
	КОНЕЦ	

В решении задачи участвуют три программы, которые описаны оператором ШАГ. Это ПРИМ1, ПРИМ2, ПРИМ3. Программы должны быть записаны с помощью средств записи программ на МЛ. Начать вычислительный процесс при данном задании можно с этикеток ВЕСНА и ВЕГА. Если начать вычислительный процесс с этикетки ВЕСНА, то будет выполнена программа ПРИМ1. Затем на ПМ будет напечатано УСТАНОВИТЬ ОЧЕРЕДНОЙ МАССИВ НА УВК и произойдет останов. После установки необходимо набрать директиву на продолжение задания. Будет выполнен шаг ПРИМ2. После программы ПРИМ2 в зависимости от результатов ее работы вычислительный процесс может быть продолжен либо с этикетки ВЕСНА, либо с этикетки ВЕГА. Если содержимое ячейки СТ1 поля СТ не совпадает ни с А, ни с В — произойдет останов вычислительного процесса. На ПМ будет напечатано НЕТ ЗНАЧЕНИЯ <имя> В ТАБЛИЦЕ. Следом печатается УКАЖИТЕ ДЕЙСТВИЕ. Продолжить вычислительный процесс можно с любой из этикеток в задании. Можно повторить участок с последней выполнившейся этикетки или окончить работу. Если значение СТ1 равно или А, или В, а соответствующая этому значению этикетка не присутствует в задании, то происходит останов вычислительного процесса и печатается на ПМ НЕТ МЕТКИ <имя>. Вслед печатается УКА-

ЖИТЕ ДЕЙСТВИЕ. После этого сообщения пользователь может выполнить одно из следующих действий (в зависимости от ответа): продолжить выполнение вычислительного процесса со следующего оператора задания; повторить участок процесса с последнего встреченного оператора, имеющего этикетку; передать управление любому оператору, имеющему этикетку; прекратить выполнение вычислительного процесса. В случае, если вычислительный процесс выходит на этикетку ВЕГА, выполняется программа ПРИМ3 и происходит останов вычислительного процесса. На ПМ печатается УКАЖИТЕ ДЕЙСТВИЕ.

3.9. ОРГАНИЗАЦИЯ И ВЕДЕНИЕ МАССИВА КАЛЕНДАРНО-ПЛАНОВЫХ НОРМАТИВОВ

В качестве примера использования описанной в этой главе системы программ рассмотрим задачу организации и ведения массива календарно-плановых нормативов.

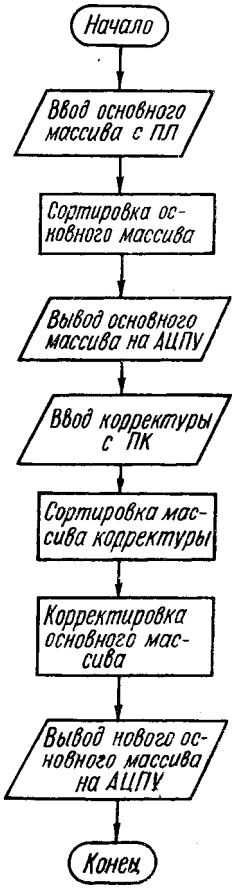


Рис. 3.17. Блок-схема решения задачи ведения календарно-плановых нормативов

В задаче используются следующие программы обработки данных:

— ввод экономической информации с перфоленты (основного массива);

— сортировка массива информации на магнитных лентах (основного массива и корректур);

— вывод экономической информации на печать (основного массива);

— ввод экономической информации с перфокарт (массива корректур);

— корректировка экономической информации на магнитных лентах.

Программы, реализующие перечисленные выше процедуры, объединяются в пакет и запускаются в работу с помощью программы управления пакетом задачи.

На рис. 3.17 приведена блок-схема решения рассматриваемой задачи.

Исходный массив состоит из документов, имеющих четыре связки (табл. 3.13).

Первая связка — операционный лист (ОЛ) содержит два реквизита: Шифр документа и Шифр цеха.

Вторая связка — УЧАСТ (уровень 1) состоит из реквизита Шифр участка.

Третья связка — ДЕТАЛ (уровень 2) содержит четыре реквизита: Шифр детали, Наименование детали, Шифр модификации и Шифр вида исполнения.

Четвертая связка — НОРМА (уровень 3) содержит четыре реквизита: Признак готовности, Номер захода в цех, Суммарный задел деталей на линии обработки, Суммарный задел готовых деталей.

Таблица 3.13

Ведомость календарно-плановых нормативов

Признак операционного листа	Шифр документа	Шифр цеха
(ОЛ)	НС-45	15

связка УЧАСТ		связка ДЕТАЛ					связка НОРМА				
Признак связки	Шифр участка	Признак связки	Шифр детали	Наименование детали	Шифр модификации	Шифр вида исполнения	Признак связки	Признак готовности	Номер захода в цех	Суммарный задел деталей на линии обработки	Суммарный задел готовых деталей
1	2	3	4	5	6	7	8	9	10	11	12
У)	10	Д)	340201201	ДЕТАЛЬ 1	1	5	Н)	0	1	100	50
								1	2	1000	500
								0	6	150	100
								1	4	150	1000
		Д)	340000607	ДЕТАЛЬ 2	10	9	Н)	0	1	150	1000
								1	4	1000	50
								0	6	4000	50
								1	2	100	1000

Реквизиты в данном массиве подчиняются следующим условиям:

Полное наименование реквизита	Имя реквизита	Тип	Нижняя граница (НГ)	Верхняя граница (ВГ)
Шифр документа	ДОКУМ	Т	5	5
Шифр цеха	ШЦ	Ц	1	999
Шифр участка	ШУ	Ц	10	99
Шифр детали	ШД	Ц	120 000 000	459 999 999
Наименование детали	НД	Т	5	15
Шифр модификации	ШМ	Ц	01	30
Шифр вида исполнения	ШВИ	Ц	1	9
Признак готовности	ПГ	Ц	0	1
Номер захода в цех	НЗ	Ц	1	6
Суммарный задел деталей	СЗДЛО	Ц	100	4000
Суммарный задел готовых деталей	СЗГД	Ц	50	1000

Реквизит ДОКУМ принимает всегда значения НС-45.

Связки удовлетворяют условиям:

Имя связки	Уровень связки	Признак связки	Границы связок	
			нижняя	верхняя
ОЛ	—	ОЛ)	—	—
УЧАСТ	1	У)	—	—
ДЕТАЛ	2	Д)	10	50
НОРМА	3	Н)	1	5

Необходимо скомпоновать массив, структура записей которого видна из приведенной ниже таблицы.

Реквизиты компонуемых записей удовлетворяют следующим условиям:

Идентификатор реквизита	Тип реквизита	Номер ячейки	Начальный разряд (символ)	Длина реквизита
ШЦ	ДЦ	1	1	3
ШУ	ДЦ	1	13	2
ШД	ДЦ	2	1	9
НД	Т	3,4	0	10
ШМ	ДЦ	5	1	2
ШВИ	ДЦ	5	9	1
ПГ	ВБ	5	13	4
НЗ	ВБ	5	17	4
СЗДЛО	ПЛ	6	—	—
СЗГД	ПЛ	7	—	—

Ввод информации осуществим с помощью ВВОДЛ.

В программе используются следующие массивы:

— массив с исходными описаниями ОПИС, отперфорированный на перфоленте;
— массив с переработанными описаниями КОНОД, записанный на магнитной ленте АААААА;
— массив с исходной информацией МАСС1, отперфорированный на перфоленте;
— массив со скомпонованными документами МАСС2 на магнитной ленте ВВВВВ (длина зоны 490 ячеек).

Описания пользователя имеют следующий вид:

ВВОД МАССИВА: МАСС1(ПЛ);
ВЫВОД МАССИВОВ: МАСС2(Ф490, МЛ-ВВВВВ);
ВВОД ДОКУМЕНТА НС-45: ОЛ, 1УЧАСТ, 2ДЕТАЛ (10,50), 3НОРМА (1,5);
СВЯЗКА ОЛ: ДОКУМ(Т,5,5), ШЦ(Ц,1,999);
СВЯЗКА УЧАСТ(У): ШУ (Ц, 10, 99);
СВЯЗКА ДЕТАЛ(Д): ШД (Ц, 120000000, 459999999), НД (Т, 5, 15), ШМ (Ц, 01, 30), ШВИ (Ц, 1, 9);
СВЯЗКА НОРМА (Н): ПГ(Ц, 0, 1), НЗ(Ц, 1.6), СЗДЛО(Ц, 100, 4000), СЗГД(Ц, 50, 1000);
ВЫВОД ДОКУМЕНТА (7) В МАССИВ МАСС2: ШЦ(ДЦ, 1, 1, 3), ШУ(ДЦ, 1, 13, 2), ШД(ДЦ, 2, 1, 9), НД(Т, 3, 0, 10), ШМ(ДЦ, 5, 1, 2), ШВИ(ДЦ, 5, 9, 1), ПГ(ВБ, 5, 13, 4), НЗ(ВБ, 5, 17, 4), СЗДЛО(ПЛ,6), СЗГД(ПЛ,7); ↑

Данные описания перфорируются на перфоленте, обрабатываются программой ОБРОД и записываются на магнитную ленту АААААА. Запуск программы ОБРОД будет производиться программой управления пакетом задачи.

Исходные документы, подготовленные на перфоленте, обрабатываются с помощью программы ВВОДЛ. Запуск ВВОДЛ производится во внешнем режиме с помощью программы управления пакетом задачи.

Для работы программы ВВОДЛ предлагается использование заказа, имеющего следующий вид:

Этикетка	КОП	Адреса и замечания
ИМНОД	КОНОД	МЛ-АААААА
ВЫП	КОНТР	
ВЫП	КОМПО	
НЭИ	ПЛ	

Результатом работы программы является массив МАСС2 скомпонованных документов (записей), записанный на магнитную ленту ВВВВВ.

Программа сортировки основного массива на магнитных лентах (рис. 3.17) реализуется одним шагом задания. Она осуществляет сортировку полученного массива по следующим реквизитам-признакам:

1) ШЦ — по возрастанию;
2) ШУ — по возрастанию;
3) ШД — по убыванию;
4) ПГ — по убыванию.

Обращение к программе МСОРТ будет следующим:

Этикетка	КОП	Адреса и замечания
	БАЗ	0
НАЧ	РЗВ	3
	ИП	МСОРТ; 3
	КА	АН; АН + 1469
	КА	АТРК; АТИ
	КА	А; Д
	ВЫХ	НАЧ; 0
МСОРТ	ОПР	МСОРТ
	НОП	
АТРК	КИ	4; 7
	КЧ	+ 114000В
	КЧ	+ 1524000В
	КЧ	— 10144000В
	КЧ	—41520000В
А	РЗВ	4
АТИ	КТ	ВВВВВ
	КЧ	0
	КЧ	1
Д	КТ	МАСС2
	НОП	
	КЧ	—0
	НОП	
	КНВУ	МЛ; 1
	КЧ	490
	НОП	
	НОП	
	БАЗ	1
А	РЗВ	1479

Указанная программа записывается в библиотеку пользователя под именем СОРТ1.

Результатом работы данной программы будет упорядоченный массив на МЛ.

Программа вывода основного массива на печать (рис. 3.17) реализуется за один шаг задания, содержащий обращение к программе подготовки информации для вывода на устройство печати и к программе вывода информации на устройство печати.

Вывод осуществляется по форме, приведенной в табл. 3.14.

Таблица 3.14

Ведомость календарно-плановых нормативов для цехов массового (поточного) производства

Шифр цеха	Шифр участка	Шифр детали	Наименование детали	Шифр модификации	Шифр вида исполнения	Признак готовности	Номер захода в цех	Суммарный задел деталей на линии обработки	Суммарный задел готовых деталей

В процессе вывода реквизиты ШЦ и ШУ должны выводиться только по изменению значений. Кроме того, по каждому цеху необходимо выводить итоги по реквизитам СЗДЛО и СЗГД.

Программа обращения к программам ПОДГ и ПЕЧАТ вместе с необходимыми описаниями будет иметь следующий вид:

Этикетка	КОП	Адреса и замечания
	БАЗ	0
ВЫВОД	РЗВ	3
	РИП	16
	ЗАКР	МЛ, 1
		МЛ, 2
		ПЧ, 1
	ЖОО	МЛ, 1
	КОСЛ5	0; СООБ1; 7
	ЖОО	МЛ, 2
	КОСЛ5	0;СООБ2;6
	ИП	ПОДГ; 1
	КА	0; ТИ
	ИП	ПЕЧАТ; 1
	КА	0; ТИП
	ОСВ	МЛ, 1

Этикетка	КОП	Адреса и замечания
		МЛ, 2
		ПЧ, 1
	ВЫХ	ВЫВОД; 0
ПОДГ	ОПР	ПОДГ
	НОП	
ПЕЧАТ	ОПР	ПЕЧАТ
	НОП	
СООБ1	КТ	УСТАН
		ОВИ М
		Л С И
		СХОДН
		ЫМ МА
		ССИВО
		М
СООБ2	КТ	УСТАН
		ОВИ М
		Л ДЛЯ
		МАССИ
		ВА С
		ТРОК
ТИ	КА	ДМЛ1; ДМЛ2
		АН1; АК1
		ЗН1; ЗК1
		ЗН; ЗК
		0; 0
		АН2; АК2
		ЗН2; ЗК2
		0; 0
		0; НП
	НОП	
	КИ	0; 0
	КА	0; Ф1
	КТ	
ТИП	КА	ДМЛ2; ДПЧ
	КА	АН2; АК2

Этикетка	КОП	Адреса и замечания
ДМЛ1	КТ	МАСС2
	НОП	
	КЧ	—0
	НОП	
	КНВУ	МЛ, 1
	КЧ	7000752В
	НОП	
	НОП	
ДМЛ2	КТ	МАСМ2
	НОП	
	КЧ	0
	НОП	
	КНВУ	МЛ, 2
	КЧ	30000740В
	НОП	
	НОП	
ДПЧ	КТ	МАСМ2
	НОП	
	НОП	
	НОП	
	КНВУ	ПЧ, 1
	КЧ	11648
	НОП	
	НОП	
Ф1	КА	0; ЗАП
	КА	Т; 0
	КИ	0; 50В
	КА	ЗАГ; ВШ
	КА	ВШ; ВШ
	КА	НРПР; 14В
	КТ	1
	НОП	
	НОП	
	КА	0; НП
	НОП	

Этикетка	КОП	Адреса и замечания
	КА	0; СТ
	КА	0; СТ1
	НОП	
	КТ	
Т	КЧ	24В
	КА	62В; Т1
	КИ	1000В; 37В
	КТ	
ЗАГ	КЧ	1
	КА	24В; Т11
	КИ	11000В; 133В
	КА	24В; ЗАГ4
	КИ	10000В; 1
	КА	27В; ЗАГ1
	КИ	0; 125В
	КА	156В; ЗАГ4
	КИ	11000В; 1
	КА	24В; Т11
	КИ	11000В; 133В
	КА	24В; ЗАГ5
	КИ	1000В; 133В
	КА	24В; ЗАГ6
	КИ	1000В; 133В
	КА	24В; ЗАГ3
	КИ	0; 12В
	КА	44В; ЗАГ4
	КИ	10000В; 1
	КА	60В; ЗАГ4
	КИ	10000В; 1
	КА	75В; ЗАГ7
	КИ	1000В; 62В
	КА	24В; ЗАГ3
	КИ	0; 12В
	КА	44В; ЗАГ4
	КИ	10000 В; 1
	КА	60В; ЗАГ4

Этикетка	КОП	Адреса и замечания
	КИ	10000; 1
	КА	75В; ЗАГ10
	КИ	0; 50В
	КА	156В; ЗАГ4
	КИ	11000В; 1
	КА	24В; Т11
	КИ	11000В; 133В
	КА	24В; ЗАГ11
	КИ	1000В; 133В
	КА	24В; Т11
	КИ	11000В; 133В
	КТ	
ВШ	КЧ	1
	КА	24В; Т11
	КИ	11000В; 133В
	КА	24В; ЗАГ11
	КИ	1000В; 133В
	КА	24В; Т11
	КИ	11000В; 133В
	КТ	
СТ	КА	0; 0
	КЧ	1000000В
	КА	0; 0
	КИ	0; 0
	КА	0; НП
	НОП	
	КА	0; Р1
		0; Р2
		0; Р3
		0; Р4
		0: Р5
		0; Р6
		0; Р7
		0; Р10
		0; Р11
		0; Р12

Этикетка	КОП	Адреса и замечания
		0; P13
		0; P14
	КТ	
CT1	КА	1; 0
	КЧ	15B
	КА	НАЯ; 15B
	КИ	0; 26B
	КЧ	777700000000B
	НОП	
	КА	0; P13
	КА	0; P14
	КТ	
P1	КЧ	0011400031B
		3401027B
		0
P2	КЧ	0151000135B
		2401037B
		0
P3	КЧ	0351000235B
		2001045B
		0
P4	КЧ	1014400301B
		11001047B
		0
P5	КЧ	2014400401B
		5000062B
		0
P6	КЧ	3014400501B
		5000067B
		0
P7	КЧ	4011000635B
		2001077B
		0
P10	КЧ	4110400741B
		1001106B
		0

Этикетка	КОП	Адреса и замечания
Р11	КЧ	4150401041В
		1001114В
		0
Р12	КЧ	4210401141В
		1001122В
		0
Р13	КЧ	5014401201В
		5001131В
		1
Р14	КЧ	6014401301В
		5001146В
		1
ЗАП	КА	0; Р1
		0; Р2
		0; Р3
		0; Р4
		0; Р5
		0; Р6
		0; Р7
		0; Р10
		0; Р11
		0; Р12
		0; Р13
	КТ	0; Р14
	КТ	ПРИМЕ
Т1		Р ИСП
		ОЛЬЗО
		ВАНИЯ
		БИБЛ
	КТ	ИОТЕК
Т11	КТ	И
ЗАГ1	КТ	—
		ВЕДОМ
		ОСТЬ
		КАЛЕН
		ДАРНО

Продолжение

Этикетка	КОП	Адреса и замечания
		— ПЛАН ОВЫХ НОРМА ТИВОВ ДЛЯ ЦЕХОВ МАСС ОВОГО (ПОТ ОЧНОГ О) ПР ОИЗВО ДСТВА
ЗАГ3	КТ	:
		:
ЗАГ4	КТ	:
ЗАГ5	КТ	: ШИФ Р : ШИФР : ШИФР : Н АИМЕН ОВАНИ Е : ШИФ Р : ШИ ФР : П РИЗ-: НОМЕР : СУМ МАРНЫ Й : СУММА РНЫЙ :

Этикетка	КОП	Адреса и замечания
ЗАГ6	КТ	: ЦЕХ А :У ЧАСТК А: ДЕТАЛ И : ДЕТ АЛИ : МОД И-:ВИ ДА :Н АК : ЗАХО- : ЗАДЕ Л ДЕТ А- :З АДЕЛ ГОТО- :
ЗАГ7	КТ	:ФИКА -:ИСП ОЛ:ГО ТОВ:Д А В : ЛЕЙ Н А ЛИН ИИ:ВЫ Х ДЕТ АЛЕЙ :
ЗАГ10	КТ	:ЦИИ :НЕН ИЯ: НО СТИ: Ц ЕХ : ОБРА БОТКИ __:__

Этикетка	КОП	Адреса и замечания
		: _ _ _ 1 : 2 : 3 : 4 _ _ _ _ _ : 5 : 6 : 7 : 8 : 9 : 10 _ _ _ _ _ :
НАЯ	КТ	ИТОГО ПО Ц ЕХУ
АН1	РЗВ	755В
АК1	РЗВ	1
ЗН1	РЗВ	9
ЗК1	РЗВ	1
ЗН	РЗВ	9
ЗК	РЗВ	1
АН2	РЗВ	743В
АК2	РЗВ	1
ЗН2	РЗВ	32В
ЗК2	РЗВ	1
НРПР	РЗВ	30В
	БАЗ	0
НП	НОП	

Продолжение

Этикетка	КОП	Адреса и замечания
	РЗВ	2
	РИП	16
	СФЗ	РАБ1; РАБ2
	ВЫХ	НП; 0
РАБ1	КЧ	1
РАБ2	КЧ	1

Процедура ввода массива корректур с перфокарт осуществляется в два шага задания:

— обработка описаний пользователя и запись на магнитную ленту описаний, пригодных для работы программы;

— ввод массива корректур с перфокарт на магнитную ленту.

Массив корректур заполняется по форме, приведенной в табл. 3.14, и перфорируется на перфокарты.

Реквизиты в данном массиве, имеющие те же идентификаторы, что и при обращении к программе ввода информации с ПЛ, должны подчиняться следующим условиям:

Идентификатор реквизита	Колонки, занимаемые реквизитом	Тип реквизита	Нижняя граница	Верхняя граница	Признак суммирования
ШЦ	1—3	Ц	1	999	С
ШУ	4—5	Ц	10	99	С
ШД	6—19	Ц	$12 \cdot 10^7$	$46 \cdot 10^7$	
НД	20—29	Т	—	—	
ШМ	30—31	Ц	1	30	
ШВИ	32	Ц	1	9	
ПГ	33	Ц	0	1	
НЗ	34	Ц	1	6	
СЗДЛО	35—38	Ц	0	4000	
СЗГД	39—42	Ц	0	1000	

В отличие от программы ввода информации с ПЛ нижние границы реквизитов СЗДЛО и СЗГД взяты равными нулю. Равенство нулю значений реквизитов СЗДЛО и СЗГД выбрано в качестве признака удаления. Поэтому документы, в которых реквизиты СЗДЛО и СЗГД принимают нулевые значения, не должны быть выброшены из массива корректур как ошибочные.

Результатом работы программы должен явиться массив корректур, имеющий ту же структуру, что и основной массив.

Операторы описания процедуры ввода представляются на бланках ССК следующим образом:

Этикетка	КОП	Адреса и замечания
		НАЧАЛО, МЛ
ОИМ	КОРРИ	80
ОКМ	КОРРМ	Ф490, МЛ-ВВВВВ
ОМ		10, 1, П-С, 50
ОР		ШЦ, Ц, 1, 3, С, 1, 999
ОР		ШУ, Ц, 4, 5, С, 10, 99
ОР		ШД, Ц, 6, 19, 120000000, 460 000 000
ОР		НД, Т, 20, 29
ОР		ШМ, Ц, 30, 31, 1
ОР		ШВИ, Ц, 32, 32, 1, 9
ОР		ПГ, Ц, 33, 33, 0, 1
ОР		НЗ, Ц, 34, 34, 1, 6
ОР		СЗДЛО, Ц, 35, 38, С 0, 4000
ОР		СЗГД, Ц, 39, 42. С, 0 1000
ОКД		КОРРМ, ДОК1, 10, 7
ОРКД	ДОК1	ШЦ, ДЦ, 0, 1, 3
ОКДР	ДОК1	ШУ, ДЦ, 0, 13, 2
ОРКД	ДОК1	ШД, ДЦ, 1, 1, 9
ОРКД	ДОК1	НД, Т, 2, 0, 10
ОРКД	ДОК1	ШМ, ДЦ, 4, 1, 2
ОРКД	ДОК1	ШВИ, ДЦ, 4, 9, 1
ОРКД	ДОК1	ПГ, ВБ, 4, 13, 4
ОРКД	ДОК1	НЗ, ВБ, 4, 17, 4
ОРКД	ДОК1	СЗДЛО, ПЛ, 5
ОРКД	ДОК1	СЗГД, ПЛ, 6 КОНЕЦ

Данные описания перфорируются на перфокарты по правилам перфорации ССК-программ и на первом шаге задания обрабатываются с помощью программы ОБРАБ. Эта программа вызывается и выполняется программой управления пакетом задачи. На втором шаге данного этапа производится ввод массива корректур

с перфокарт с помощью программы ВВОДК. Запуск программы ВВОДК производится во внешнем режиме с помощью программы управления пакетом задачи. В этом случае структура управляющей карты, необходимой для работы программы ВВОДК, будет иметь следующий вид:

Колонки	Значения	Пояснения
2—3	ПК	Исходный массив на ПК
4	1	Нужен контроль
5	0	Промежуточная запись не нужна
6—8	100	Необходима компоновка 1-го массива
9—10	МЛ	Машинные описания на МЛ
11—15		Имя массива
16—20		Имя магнитной ленты

Процедура сортировки корректур осуществляется так же, как и сортировка массива.

Процедура корректировки реализуется в один шаг задания и предусматривает вставку, удаление и замену документов исходного массива информации.

Реквизитами-признаками являются: «шифр цеха», «шифр участка», «шифр деталей», «признак готовности».

Как было указано выше, в качестве признака удаления используем равенство нулю значений реквизитов СЗДЛО и СЗГД. Предусмотрим в программе возможность группового удаления записей. Для выполнения описанной работы обращение к программе КОРЕК будет следующим:

Этикетка	КОП	Адреса и замечания
	БАЗ	0
НАЧАЛ	РЗВ	3
	ИП	КОРЕК; 1
	КА	0; КРАБ
	ВЫХ	НАЧАЛ; 0
КРАБ	КТ	ВУЗКГ
	КА	7; АОКР

Этикетка	КОП	Адреса и замечания
	КА	0; АПУ
	КА	АОМС; АТАС
	КА	АОМК; АТАК
	КА	АОМН; АТАН
—	КА	0; 0
	КА	0; 0
	КА	0; 0
АОКР	КЧ	4
	КТ	ДЦВ
	КЧ	10014В
	КТ	ДЦВ
	КЧ	150010В
	КТ	ДФУ
	КЧ	1000000В
	КТ	ВБУ
	КЧ	4150004В
АПУ	КИ	5; 2
	КЧ	—777777777777В
	КЧ	—777777777777В
	НОП	
	НОП	
АОМС	КТ	МАСС2
	НОП	
	КЧ	— 0
	НОП	
	КНВУ	МЛ, 1
	КЧ	7000752В
	НОП	
	НОП	
АОМК	КТ	МАСС2
	НОП	
	КЧ	— 0
	НОП	
	КНВУ	МЛ, 2
	КЧ	7000752В
	НОП	

Этикетка	КОП	Адреса и замечания
	НОП	
АОМН	КТ	МАСС2
	НОП	
	КЧ	— 0
	НОП	
	КНВУ	МЛ, 3
	КЧ	7000752В
	НОП	
	НОП	
АТАС	КА	АС; АС + 492
	КА	АЗС; АЗС + 9
	КА	АЗС1; АЗС1 + 9
АТАК	КА	АК; АК + 492
	КА	АЗК; АЗК + 9
	КА	АЗК1; АЗК1 + 9
АТАН	КА	АН; АН + 492
	КА	АЗН; АЗН + 9
КОРЕК	ОПР	КОРЕК
	НОП	
	БАЗ	2
АС	РЗВ	493
АЗС	РЗВ	10
АЗС1	РЗВ	10
АК	РЗВ	493
АЗК	РЗВ	10
АЗК1	РЗВ	10
АН	РЗВ	493
АЗН	РЗВ	10

В пакет задачи включаются следующие шаги задания:

Назначение шага задания	Имя программы, реализующей шаг задания	Этикетка шага
Обработка описания процедуры ввода основного массива с ПЛ	ОБРОД	ВВОДО
Ввод основного массива с ПЛ	ВВОДЛ	
Сортировка основного массива	СОРТ1	СОРТО

Назначение шага задания	Имя программы, реализующей шаг задания	Этикетка шага
Вывод основного массива на АЦПУ	ВЫВОД	ВЫВОД
Обработка описанной процедуры ввода массива корректур с ПК	ОБРАБ	ВВОДК
Ввод массива корректур с ПК	ВВОДК	
Сортировка массива корректур	СОРТ2	СОРТК
Корректировка основного массива	КОР1	КОРЕК

Задание на реализацию вычислительного процесса будет следующим:

Этикетка	КОП	Адреса и замечания
	ЗАДАЧ	КОНПР
ВВОДО	ШАГ	ОБРОД
	ШАГ	ВВОДЛ
СОРТО	ШАГ	СОРТ1
ВЫВОД	ШАГ	ВЫВОД
ВВОДК	ШАГ	ОБРАБ
	ШАГ	ВВОДК
СОРТК	ШАГ	СОРТ2
КОРЕК	ШАГ	КОР1
	КОНЕЦ	

Программой КОНТР это задание приводится к виду, приемлемому для работы программы управления вычислительным процессом ДИСП. Обработанное задание записывается на МЛ с идентификатором КОНПР.

С помощью программы ДИСП производится запуск задачи КОНПР. Начать вычислительный процесс можно с любой программы, описание которой имеет этикетку. Останов вычислительного процесса будет происходить перед выполнением очередного управляющего оператора, имеющего этикетку. При этом на ПМ печатается УКАЖИТЕ ДЕЙСТВИЕ. Напечатав на ПМ нужную этикетку, можно продолжить вычисление с любого оператора, имеющего этикетку. Если процесс нужно продолжать со следующего шага, то достаточно дать ответ: *№◇. Например, после выполнения программ ОБРОД и ВВОДЛ можно распечатать полученный массив.

Глава 4

АЛГОРИТМИЧЕСКИЙ ЯЗЫК КОБОЛ

4.1. ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ КОБОЛ

Алгоритмический язык КОБОЛ является процедурно-ориентированным языком программирования задач обработки данных. В автоматизированных системах управления задачи такого типа являются наиболее распространенными. Например, в АСУ Павлодарским тракторным заводом такие задачи, как планирование и управление технической подготовкой производства нового изделия методом сетевых графиков, составление плана потребности материалов и покупных изделий, расчет плана по труду и заработной плате, составление калькуляций нормативной себестоимости, расчет стоимости основных фондов, платы за фонды и амортизационных отчислений, определение плановой себестоимости продукции, учет готовой продукции, движения материалов, покупных изделий и инструмента на заводе, учет труда и заработной платы, бухгалтерский учет брака, составление подетальных планов производства, расчет загрузки оборудования, учет хода производства, межцеховое оперативно-календарное планирование, внутрицеховое оперативно-календарное планирование, внутрицеховой учет хода производства, простоев оборудования и брака, реализованы с помощью языка КОБОЛ. Отметим, что программы, обеспечивающие организацию и ведение массивов нормативно-справочной информации, составлены на этом же языке. В перечисленных задачах, являющихся типичными представителями задач АСУ, вычислений меньше, чем всевозможных перемещений данных, выполняются они над большими входными массивами, соизмеримыми с объемом выходных результатов. При решении подобных задач форматы входных и выходных данных обычно предварительно определены во всех подробностях и программист должен организовать ввод и вывод данных, строго следуя заданным форматам. Более того, входные данные зачастую представляют собой уже существующие массивы, хранимые либо на магнитной ленте, либо на перфокартах, либо на перфолентах, и программисту необходимо приспособиться к их форме.

В данной главе описываются основные конструкции языка КОБОЛ, транслятор с которого реализован на ЭВМ «Минск-32»*. Он разработан на основе языка системы автоматической обработки данных (САОД) для ЭВМ «Минск-22» [10].

Основные отличия языка КОБОЛ для ЭВМ «Минск-32» от входного языка САОД для ЭВМ «Минск-22» состоят в следующем:

расширен алфавит языка в основном за счет латинских букв, не совпадающих по написанию с русскими;

увеличена разрядность чисел с 10 до 18 цифр;

допускается 2-й уровень индексации;

допускаются сложные условия;

введен глагол СОРТИРОВАТЬ;

введены средства межпрограммной связи;

обеспечена возможность выделения подпрограмм в программе задачи.

Программа, записанная на языке КОБОЛ, состоит из четырех отдельных частей, называемых разделами. Они записываются в том порядке, в котором их названия перечислены ниже:

раздел идентификации,

раздел оборудования,

раздел данных,

раздел процедур.

В разделе идентификации указываются: название программы, фамилия ее автора, дата написания и другие сведения, необходимые для ведения документации. Обычно на каждой ЭВМ существует своя система ведения документации.

В разделе оборудования определяются ЭВМ, на которой будет производиться трансляция КОБОЛ-программы, и ЭВМ, на которой будет производиться счет по созданной транслятором объектной программе. В этом же разделе определяются внешние устройства, на которых будет располагаться каждый из массивов, и таким образом определяется оборудование, необходимое для решения данной задачи.

В разделе данных описываются форматы входных и выходных данных, подлежащих обработке, и способы организации данных в массиве. По своему смыслу описание характеризует изображение данного на листе бумаги, а не способ размещения его в памяти машины, а именно: описывается, из каких знаков данное составлено (буквы, цифры и т. д.), сколько цифр содержится в числе, какова последовательность колонок в некоторой таблице, сколько в ней строк, каков способ редактирования колонки, каково содержание колонок к моменту решения задачи и др. В этом же разделе описываются особенности хранения записей

---

* В разработке этого варианта языка участвовали Л. М. Романовская, Т. А. Савченко, Л. С. Фельдман.

в массивах и структура всех рабочих областей, которые могут потребоваться.

В разделе процедур описываются действия, выполняемые над данными. Действия указываются с помощью специальных операторов.

Подобно любому языку, язык КОБОЛ имеет свой алфавит, составляющие которого будем называть литерами. В алфавите КОБОЛа 70 литер. Программа, записанная на КОБОЛе, не может содержать других значков, кроме литер, составляющих алфавит КОБОЛа.

Литеры КОБОЛа заключают в себе все знаки русского алфавита, кроме литер Ъ, Ё, те латинские буквы, которые отличаются по написанию от русских (D, F, G, I, J, L, N, Q, R, S, U, V, W, Z), цифры от 0 до 9 и специальные литеры.

Ниже приводится список специальных литер.

Литера	Название
␣	пробел
+	знак плюс
−	знак минус (дефис)
*	знак умножения (звездочка)
/	знак деления (черта)
=	знак равенства
,	запятая
;	точка с запятой
.	точка и десятичная точка *
'	открывающая кавычка
'	закрывающая кавычка
(	левая (открывающая) скобка
)	правая (закрывающая) скобка
>	больше
<	меньше

Перечисленные литеры и только они используются для записи КОБОЛ-программ.

Будем считать, что пропуск между словами обозначается с помощью специальной литеры, которую будем называть пробелом. Иногда пробел будет обозначаться символом «␣». Исключая случай нечисловых литералов, два или большее число пробелов подряд рассматриваются как один. Многократные пробелы могут быть использованы для того, чтобы сделать исходную КОБОЛ-программу более читабельной, однако они не оказывают никакого влияния на объектную программу.

Следует отметить, что данные, подлежащие обработке, могут состоять из любых литер, имеющихся в наборе вычислительной

---

* В КОБОЛе вместо десятичной запятой употребляется десятичная точка.

машины, т. е. в значении данных могут встречаться литеры, не принадлежащие к алфавиту КОБОЛа.

Из литер строятся слова, которые по правилам синтаксиса данного языка объединяются во фразы, статьи, выражения, операторы, предложения, параграфы, секции, процедуры и разделы. Слово в КОБОЛе может состоять не более чем из тридцати литер, каждая из которых выбирается из знаков русского и латинского алфавитов, цифр и дефиса. Дефис не может быть первым или последним символом в слове. Одним из разделителей слов в КОБОЛ-программе является пробел. Однако в основных конструкциях рассматриваемого варианта языка КОБОЛ на понятие слова накладывается еще одно ограничение — первым символом слова должна быть буква. Слова, состоящие только из цифровых символов, используются при образовании номера уровня, числовых литералов и строк литер шаблона.

Слова КОБОЛа можно разделить на две основные группы: служебные слова и слова пользователя.

Служебные слова — это собственно словарь КОБОЛа, т. е. фиксированный набор слов, имеющих строго определенный смысл. Они заранее установлены, имеют специальное значение для транслятора, используются в соответствии со специальными требованиями КОБОЛа и не могут быть употреблены в качестве слов пользователя. Имеются две разновидности служебных слов: основные и вспомогательные.

Основные слова — это служебные слова, которые несут определенную смысловую нагрузку в КОБОЛ-программе и наличие которых обязательно, если конструкции, в которых имеются эти слова, используются в исходной программе.

Вспомогательные слова — это служебные слова, которые могут быть опущены и используются только для удобства при чтении. Наличие или отсутствие любого вспомогательного слова не влияет на трансляцию, однако неправильное его написание недопустимо.

Все остальные слова определяются пользователем и называются словами пользователя. В качестве слов пользователя не могут выступать служебные слова. Например, в конструкциях:

ДЛЯ ОСНОВНОЙ — МАССИВ ПРЕДНАЗНАЧИТЬ МЛ 3 СО СМЕНОЙ КАТУШЕК.

ВЫЧИСЛИТЬ ИТОГ=ИТОГ+СУММА ПРИ ПЕРЕПОЛНЕНИИ ВЫПОЛНИТЬ ПРОВЕРКА

слова ПРЕДНАЗНАЧИТЬ, СМЕНОЙ, ВЫЧИСЛИТЬ, ПЕРЕПОЛНЕНИИ, ВЫПОЛНИТЬ являются основными, слова ДЛЯ, СО, КАТУШЕК, ПРИ — вспомогательными, а слова ОСНОВНОЙ-МАССИВ, 3, ИТОГ, СУММА, ПРОВЕРКА — словами пользователя. Слова пользователя делятся на три группы: названия, литералы и шаблоны.

Названия подразделяются на названия данных, индексов, процедур, условные и названия специальных данных в зависимо-

сти от того, что они именуют. Названия образуются по правилам образования слов. Названия должны быть уникальными, другими словами, различным объектам исходной КОБОЛ-программы должны соответствовать различные названия.

Названия специальных данных это зарезервированные слова СЧЕТЧИК и ТЕКУЩАЯ-ДАТА. Слово СЧЕТЧИК является названием ячейки оперативной памяти, которая служит для хранения информации, получающейся при использовании оператора ПРОСМОТРЕТЬ. ТЕКУЩАЯ-ДАТА используется в операторе ПОМЕСТИТЬ и содержит текущую дату (год, месяц, число), занесенную с помощью манипуляций оператора ЭВМ. Эти данные в разделе данных не описываются.

Литералами называют константы, представляющие любое число или величину, к которым обращаются по значению, а не по имени. Например, если имеется ПИ=3.14, то к числу 3.14 можно обращаться, по имени ПИ. Однако это число можно использовать, непосредственно задавая его значение. В записи A1=A2+ +3.14 число 3.14 — литерал. Будем различать два типа литералов — нечисловые и числовые.

Нечисловой литерал заключается в кавычки и содержит максимум 128 любых литер из набора литер машины, кроме кавычек. Кавычки в размер литерала не включаются.

Например:

'СПИСОК СОТРУДНИКОВ'
''
'877'
'АВГУСТ 16, 1938'

Числовой литерал образуется путем написания строки, состоящей не более чем из 18 цифр; допускается включение одной десятичной точки. Точка не может быть крайней правой литерой литерала. Если точка не записана, то предполагается, что литерал — целое число. Числу может предшествовать знак плюс или минус. Если перед числом знак отсутствует, то литерал считается положительным. Между знаком литерала и первой цифрой не должно быть пробела.

Например:

19876
—.000001
+749.0072
—843.1

Литерал, которому присвоено название, называется именуемой константой.

Именуемая константа описывается в разделе данных, где ей присваивается название и указывается ее значение. Именуемая константа используется точно так же, как любая другая величина. Например, может понадобиться неоднократное использование

числа 3.1415926. Присвоив в разделе данных этой константе название, например, ПИ, в разделе процедур можно пользоваться этим названием вместо самой константы.

Некоторым именуемым константам присвоены постоянные названия. Они автоматически распознаются транслятором, и нет необходимости определять их в разделе данных. Эти константы называются фигуральными (стандартными).

В языке допускаются следующие фигуральные константы:

НУЛЬ — представляет величину 0 или последовательность нулей;

ПРОБЕЛ — представляет последовательность пробелов;

КАВЫЧКА — представляет последовательность открывающих кавычек (');

ВСЕ нечисловой-литерал — представляет последовательность символов, указанных литералом.

Количество нулей, пробелов или кавычек определяется размером соответствующего поля. Аналогично количество повторений символов, указанных за словом ВСЕ, определяется размером поля, которое заполняется указанными в кавычках символами. Например, запись вида ВСЕ '9' соответствует одной или нескольким последовательным цифрам 9. Из описаний конструкций языка будет ясно, где допускается использование фигуральных констант.

Все слова в программе, как правило, должны заканчиваться пробелом. Пробел считается границей слова, после которой начинается новое слово. Так, например, строка литер

ЧИТАТЬ МАССИВ — ЛИЦЕВЫХ — СЧЕТОВ

состоит из двух слов (ЧИТАТЬ и МАССИВ — ЛИЦЕВЫХ — СЧЕТОВ), хотя второе слово привычнее рассматривать как три разных слова. Однако транслятор считает словом последовательность литер до пробела, поэтому строка МАССИВ — ЛИЦЕВЫХ — СЧЕТОВ воспринимается им как одно слово.

В КОБОЛ-программе отдельные фразы и предложения разделяются знаками препинания, в качестве которых используются: «.», «,» и «;». Подобно литере пробела, знаки препинания, а также открывающая и закрывающая скобки служат границей слова. Количество пробелов между словами произвольно. Места, где могут быть поставлены запятая и точка с запятой, указываются в форматах.

Если эти знаки не используются, вместо них должен быть по крайней мере один пробел.

Для записи программ используется специальный бланк (рис. 4.1). Каждая строка бланка затем перфорируется на карту, при этом номер позиции на бланке соответствует номеру колонки на перфокарте. Позиции 1÷5 (Идентиф) занимает идентификатор программы, представляющий сокращенное название этой программы. В позициях 6÷8 (Лист) размещается номер

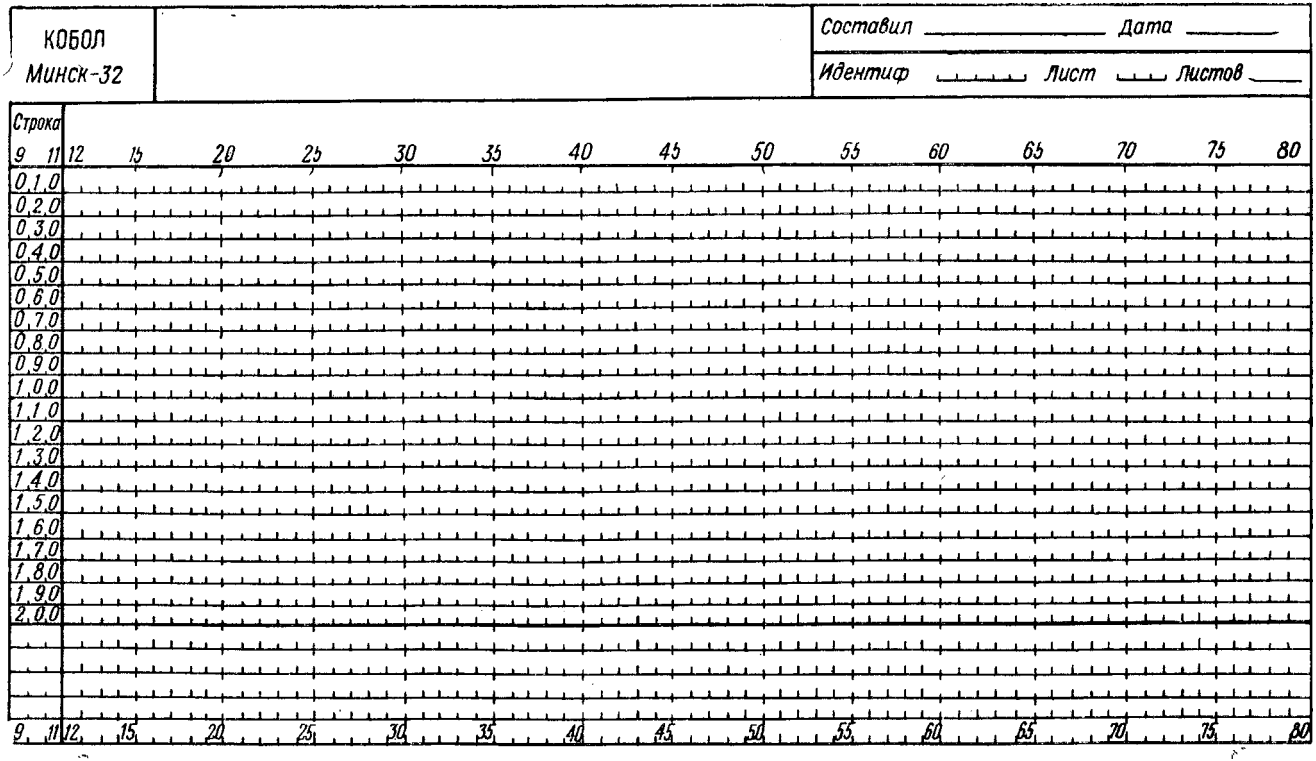


Рис. 4.1. Бланк для записи КОБОЛ-программы

листа программы, в позициях 9÷11 (Строка) — номер строки. Позиции 12÷80 отведены под текст программы. Нумерация строк нанесена непосредственно на бланк, причем для удобства внесения изменений строки пронумерованы через 10. Последние пять строк отведены для написания изменений. 12-я позиция служит для вспомогательных целей, обычно она пустая, при необходимости в ней размещаются признаки того, что слово, начатое на предыдущей строке, продолжается на данной (таким признаком служит дефис) или что данная строка является отладочной (таким признаком служит звездочка).

При описании синтаксиса языка КОБОЛ будем пользоваться специальной системой обозначений. Если слова записаны прописными буквами, то это служебные слова, которые должны быть записаны, как они указаны. Если служебные слова являются основными, то их наличие в КОБОЛ-программе обязательно в тех случаях, когда выбрана определенная альтернатива. Вспомогательные служебные слова могут быть опущены, если же они записаны, то не играют никакой роли при трансляции. Назначение этих слов состоит в том, чтобы сделать исходную программу более читабельной. Для простоты вспомогательные слова при описании конструкций языка подчеркнуты так же, как и в приведенном ниже списке служебных слов языка.

Слова, написанные строчными буквами, должны быть заменены словами пользователя.

Отметим, что в приводимых ниже примерах используются только литеры алфавита КОБОЛа, поэтому строчные буквы, подчеркивание и другие средства описания синтаксиса языка, естественно, не применяются.

Многие элементы КОБОЛа являются необязательными. Употребление в формате квадратных скобок обозначает, что заключенная в скобки часть формата может быть использована или опущена по желанию пользователя. Иногда программист имеет возможность сделать выбор. Это указывается написанием различных альтернатив в фигурных скобках. Фигурные скобки, употребленные в формате, обозначают, что пользователь должен выбрать один из предлагаемых вариантов формата, содержащихся в скобках. Если информация, заключенная в квадратные скобки, заканчивается многоточием, это означает, что соответствующая часть формата может быть неоднократно повторена.

Ниже приведен список служебных слов языка КОБОЛ, транслятор с которого реализован на ЭВМ «Минск-32».

Назначение этих слов и конструкции, в которых их можно (или нужно) использовать, будут приведены в описании форматов языка. В качестве слов пользователя приведенные в списке слова применяться не могут.

В списке служебных слов и описанных форматов языка вспомогательные слова подчеркнуты.

АВТОР
БЕЗ
БЛОКЕ
БОЛЬШЕ
БУКВЕННОЕ
В
ВВОДА-ВЫВОДА
ВЕДУЩИЕ
ВК
ВЛ
ВОЗРАСТАНИЮ
ВОЙТИ
ВСЕ
ВХОДНОЙ
ВЫВОДА
ВЫДАТЬ
ВЫЗВАТЬ
ВЫЙТИ
ВЫПОЛНИТЬ
ВЫХОДНОЙ
ВЫЧ
ВЫЧИСЛЕНИЙ
ВЫЧИСЛИТЬ
ГОДНОСТИ
ДАННЫХ
ДАТА-НАПИСАНИЯ
ДЛЯ
ДО
ДОПОЛНИТЕЛЬНЫХ
ЕСЛИ
ЕСТЬ
ЗАВИСИМОСТИ
ЗАКРЫТЬ
ЗАМЕНЯЯ
ЗАМЕЧАНИЯ
ЗАП
ЗАПИСЕЙ
ЗАПИСЬ
ЗАПОЛНИТЕЛЬ
ЗАТЕМ
ЗНАЧ
ЗНАЧЕНИЕ
ЗОН
ЗОНА
И
ИДЕНТИФИКАЦИИ
ИЗМЕНИТЬ
ИЗ-ПРОГРАММЫ
ИЛИ
ИНАЧЕ
ИНДЕКСА
ИНДЕКСИРУЕТСЯ
ИСПОЛЬЗУЯ
ИСХОДНАЯ-МАШИНА
К
КАВЫЧКА
КАТУШЕК
КАТУШКИ
КАТУШКЕ
КЛЮЧА
КОГДА
КОНФИГУРАЦИИ
КОНЦА
КОНЦЕ
ЛИСТОВ
ЛЫ
МАССИВА
МАССИВОВ
МЕНЬШЕ
МЕНЯЯ
МЕТКИ
МЕТОД
МИНСК-32
МЛ
НА
НЕ
НУЛЬ
ОБОРУДОВАНИЯ
ОБЩАЯ
ОДНОЙ
ОКРУГЛЯЯ
ОМ
ОПУЩЕНЫ
ОС
ОСВОБОДИТЬ
ОСВОБОЖДЕНИЕМ
ОСТАНОВИТЬ
ОТ
ОТКРЫТЬ
ПАМЯТИ
ПЕРЕОПРЕДЕЛЯЕТ
ПЕРЕЙТИ
ПЕРЕМОТКИ
ПЕРЕПОЛНЕНИИ
ПЕРЕПРОГОН
ПЕРВОГО
ПЕРВЫЙ
ПЕРЕХОДА
ПЕРИОД
ПИСАТЬ
ПЛОТНЫЙ
ПОСЛЕ
ПО
ПОВТОРЯЕТСЯ
ПОЛУЧАЯ
ПОМЕСТИТЬ
ПРЕДЛОЖЕНИЕ
ПРЕДНАЗНАЧИТЬ
ПРИ
ПРИМЕЧАНИЕ
ПРИНЯТЬ
ПРОБЕЛ
ПРОГРАММА
ПРОДВИГАЯ
ПРОСМОТРЕТЬ
ПРОЦЕДУР
ПЧ
РАБОТУ
РАБОЧАЯ-МАШИНА
РАБОЧЕЙ-ПАМЯТИ
РАВНО
РАЗ
РАЗА
РАЗДЕЛ
РАЗМЕР
РЕЗЕРВИРОВАТЬ
С
СВЯЗИ
СЕКЦИЯ
СЛЕДУЮЩЕЕ
СМЕНОЙ
СО
СОРТИРОВАТЬ
СПЕРВА
СПЕЦИАЛЬНЫЕ-НАЗВАНИЯ
С-РАЗДЕЛИТЕЛЯМИ
СТАНДАРТ
СТАНДАРТНЫ
СТРОК
СТРОКИ
СТРОКУ
СЧЕТЧИК
СЧИТАЯ
УБЫВАНИЮ
УПРАВЛЕНИЕ-ВВОДОМ-ВЫВОДОМ
УПРАВЛЕНИЕ МАССИВАМИ
УСТАНОВИТЬ
ЦИФРОВОЕ
ЧИТАТЬ
Ш
ШАБЛОН
ШИФР
ЫК
ЫЛ

4.2. РАЗДЕЛ ИДЕНТИФИКАЦИИ

Раздел идентификации используется для приведения справочной информации о программе. С этого раздела должна всегда начинаться программа на КОБОЛе. Раздел идентификации имеет следующий формат:

РАЗДЕЛ ИДЕНТИФИКАЦИИ.
ПРОГРАММА. комментирующее-предложение
[АВТОР. комментирующее-предложение]
[ДАТА-НАПИСАНИЯ. комментирующее-предложение]
[ЗАМЕЧАНИЯ. комментирующее-предложение]

Первые две строки обязательны. Параграф ПРОГРАММА записывается первым после заголовка раздела. Остальные параграфы, если они присутствуют, могут следовать за ним в любом порядке.

Комментирующее-предложение есть любая комбинация слов языка, заканчивающаяся точкой. Эти предложения не влияют на процесс трансляции, они воспроизводятся в листах программы.

Приведем пример раздела идентификации:

РАЗДЕЛ ИДЕНТИФИКАЦИИ.
ПРОГРАММА. УЧЕТ ОСНОВНЫХ СРЕДСТВ.
АВТОР. ИВАНОВА НИНА МИХАЙЛОВНА.
ДАТА-НАПИСАНИЯ. 10 НОЯБРЯ 1971 Г.
ЗАМЕЧАНИЯ. ПРИГОДНА ТОЛЬКО ДЛЯ ВНУТРИЗАВОДСКИХ РАСЧЕТОВ.

4.3. РАЗДЕЛ ОБОРУДОВАНИЯ

Раздел оборудования следует за разделом идентификации и начинается с заголовка РАЗДЕЛ ОБОРУДОВАНИЯ. Этот раздел предназначен для приведения справочной информации о машине, на которой будет транслироваться исходная программа, о машине, на которой будет выполняться полученная рабочая программа, и указаний по распределению оперативной памяти, памяти на магнитных лентах и другого оборудования вычислительной машины, которое будет использоваться в программе.

Раздел оборудования состоит из двух секций: конфигурации и ввода-вывода, которые в свою очередь подразделяются на параграфы. Параграфы состоят из статей. Каждая статья содержит одну или несколько фраз и заканчивается точкой.

Секция конфигурации состоит из трех параграфов: ИСХОДНАЯ-МАШИНА, РАБОЧАЯ-МАШИНА и СПЕЦИАЛЬНЫЕ-НАЗВАНИЯ, из которых обязательным является только параграф РАБОЧАЯ-МАШИНА.

Формат секции конфигурации:

СЕКЦИЯ КОНФИГУРАЦИИ.

[ИСХОДНАЯ-МАШИНА. МИНСК-32.]
РАБОЧАЯ-МАШИНА. МИНСК-32 [; РАЗМЕР ПАМЯТИ целое ЛИСТОВ].

[СПЕЦИАЛЬНЫЕ-НАЗВАНИЯ. название-устройства-1 ЕСТЬ условное-название-1.
[название-устройства-2 ЕСТЬ условное-название-2...]

В параграфе ИСХОДНАЯ-МАШИНА, если он присутствует, называется ЭВМ «Минск-32» как машина, для которой составлена и на которой будет транслироваться исходная программа.

В параграфе РАБОЧАЯ-МАШИНА называется ЭВМ «Минск-32» как машина, на которой будет выполняться после трансляции рабочая программа, и указывается размер оперативной памяти, отведенной для рабочей программы. В этом параграфе целое должно быть таким, чтобы в сумме с длиной резидентной части программы ДИСПЕТЧЕР оно не превышало объема оперативной памяти машины. Однако параграф содержит информацию только для программиста, используется для документации и на трансляцию не влияет.

Параграф СПЕЦИАЛЬНЫЕ-НАЗВАНИЯ используется для присвоения внешним устройствам машины условных названий, которые используются в операторе ВЫДАТЬ. Формат параграфа:

СПЕЦИАЛЬНЫЕ-НАЗВАНИЯ.
название-устройства-1 ЕСТЬ условное-название-1.
[название-устройства-2 ЕСТЬ условное-название-2...]

В качестве названий устройств могут использоваться ПЧ целое, где целое может изменяться от 1 до 63. Каждому из устройств может быть присвоено только одно условное название. Если нет необходимости давать условные названия, этот параграф опускается.

Секция ввода-вывода состоит из двух параграфов УПРАВЛЕНИЕ-МАССИВАМИ и УПРАВЛЕНИЕ-ВВОДОМ-ВЫВОДОМ.

Параграф УПРАВЛЕНИЕ-МАССИВАМИ используется для назначения массивам условных номеров устройств (типа устройства и номера в типе). Машинные номера устройств назначаются ДИСПЕТЧЕРом при выполнении рабочей программы.

Параграф УПРАВЛЕНИЕ-МАССИВАМИ представляет собой последовательность фраз ПРЕДНАЗНАЧИТЬ. Формат этой фразы следующий:

УПРАВЛЕНИЕ-МАССИВАМИ.
ДЛЯ название-массива ПРЕДНАЗНАЧИТЬ название-устройства-1 [, название-устройства-2] [; СО СМЕНОЙ КАТУШЕК] [; РЕЗЕРВИРОВАТЬ целое ДОПОЛНИТЕЛЬНЫХ ЗОН].

Фраза ПРЕДНАЗНАЧИТЬ используется для перечисления устройств, предназначенных названному в этой фразе массиву. Фраза является обязательной для каждого массива, используемого при решении задачи.

В качестве названия устройства может быть указано одно из следующих:

МЛ целое — магнитная лента;

ВЛ целое — устройство ввода с перфоленты;

ЫЛ целое — устройство вывода на перфоленту;

ВК целое — устройство ввода с перфокарт;

ЫК целое — устройство вывода на перфокарты;

ПЧ целое — устройство вывода на широкую печать.

Отметим, что МЛ, ВЛ, ЫЛ, ВК, ЫК, ПЧ являются основными служебными словами рассматриваемой версии языка КОБОЛ. Целое может принимать значения от 1 до 63.

Все устройства, предназначенные для массива, должны быть перечислены в одной фразе и соответствовать методу записи массива. Массиву не должно назначаться несколько устройств одного типа. Исключение представляет сортируемый массив, для которого требуется две МЛ. Условные номера устройств, назначенные сортируемому массиву, нельзя назначать другим массивам, обрабатываемым в данной задаче после сортировки. Следующие сочетания устройств разного типа могут быть предназначены одному и тому же массиву: ЫЛ, ВЛ и ЫК, ВК, причем использоваться может сначала устройство вывода, а затем ввода. Массивам, находящимся на одной катушке, должно быть назначено одно и то же устройство.

Если массив будет располагаться на нескольких катушках, надо использовать фразу СО СМЕНОЙ КАТУШЕК. При выполнении рабочей программы оператору ЭВМ будет выдано сообщение в тот момент, когда необходима смена лент, и оператор должен будет поставить на указанное устройство новую ленту.

Фраза РЕЗЕРВИРОВАТЬ может быть употреблена в статье только для сортируемого массива. По этой фразе транслятор зарезервирует указанное количество зон ввода (вывода) вдобавок к основной зоне ввода (вывода) массива. Размер каждой зоны равен размеру блока массива. Целое во фразе должно быть равно $2+3К$, где $К=0,1,....$ Если фраза опущена, резервируются 2 дополнительные зоны.

Примеры.

ДЛЯ УГЛОВЫЕ-СПЕЦИФИКАЦИИ ПРЕДНАЗНАЧИТЬ МЛ 2; СО СМЕНОЙ КАТУШЕК.

ДЛЯ ОБОРОТНАЯ-ВЕДОМОСТЬ ПРЕДНАЗНАЧИТЬ ПЧ 1.

ДЛЯ ОСНОВНОЙ-МАССИВ ПРЕДНАЗНАЧИТЬ МЛ 4, МЛ 5; РЕЗЕРВИРОВАТЬ 8 ДОПОЛНИТЕЛЬНЫХ ЗОН.

Параграф УПРАВЛЕНИЕ-ВВОДОМ-ВЫВОДОМ предназначен для указания точек, в которых при выполнении рабочей программы должна производиться контрольная запись содержимого оперативной памяти, для указания массивов, которые могут иметь одну и ту же зону ввода (вывода), и перечисления массивов, расположенных на одной и той же катушке магнитной ленты.

Формат параграфа:

УПРАВЛЕНИЕ-ВВОДОМ-ВЫВОДОМ.

[ПЕРЕПРОГОН [НА ЛЫ] ПОСЛЕ {КОНЦА КАТУШКИ / целое ЗАПИСЕЙ} название-массива-1 . . . .] [ОБЩАЯ ЗОНА ДЛЯ название-массива-2, название-массива-3 [, название-массива-4 . . .] . . . .] [НА ОДНОЙ КАТУШКЕ название-массива-5, название-массива-6 [, название-массива-7 . . .] . . .]].

Параграф УПРАВЛЕНИЕ-ВВОДОМ-ВЫВОДОМ не является обязательным, но если употреблен заголовок параграфа, то по крайней мере одна из фраз должна появиться вслед за ним.

Использование фразы ПЕРЕПРОГОН позволяет при выполнении рабочей программы производить выгрузку памяти на заданное внешнее устройство в определенных контрольных точках с тем, чтобы в случае сбоев вычислителя или внешних устройств можно было, восстановив содержимое памяти и положение носителей с входной-выходной информацией, повторить прогон программы не с самого начала, а с некоторой контрольной точки.

Вариант ПОСЛЕ КОНЦА КАТУШКИ может быть определен только для массивов на перфоленте и магнитной ленте, а ПОСЛЕ целое ЗАПИСЕЙ — для массивов на магнитной ленте, перфокартах или выдаваемых на печать.

Если использован вариант ПОСЛЕ КОНЦА КАТУШКИ, контрольная выгрузка памяти будет производиться после завершения обработки очередной катушки массива, название которого указано в этой фразе, а если вариант ПОСЛЕ целое ЗАПИСЕЙ — после чтения (записи) указанного количества записей соответствующего массива. Контрольные точки должны задаваться программистом так, чтобы в момент выгрузки памяти не было массивов на перфоленте, не обработанных до конца катушки. Сортируемый массив не может использоваться в этой фразе. Например, ПЕРЕПРОГОН НА ЛЫ ПОСЛЕ КОНЦА КАТУШКИ НАРЯДЫ-НА-ПЛ.

Фраза ОБЩАЯ ЗОНА может применяться программистом для эффективного использования оперативной памяти ЭВМ за счет совмещения зон (буферов) ввода-вывода двух или более массивов. Чтобы при этом исключить наложение информации, никакие два массива, одновременно участвующие в обработке, в одной фразе встречаться не должны. Для этого достаточно, чтобы встречающиеся в одной фразе массивы не были одновременно открыты. Отметим, что для записанных в этой фразе массивов будут совмещены и зоны для записей (сегментов буферов). Все массивы, использующие одну и ту же зону ввода-вывода, обязательно должны быть перечислены в одной фразе

Название сортируемого массива не может быть употреблено в фразе ОБЩАЯ ЗОНА.

Пример.

ОБЩАЯ ЗОНА ДЛЯ ОСНОВНОЙ-МАССИВ, ЦЕННИК-МАТЕРИАЛОВ.

Фраза НА ОДНОЙ КАТУШКЕ требуется, если несколько массивов находятся на одной и той же магнитной ленте. В фразе должны быть перечислены только те массивы, которые участвуют в обработке, независимо от того, сколько массивов находится фактически на этой ленте. Порядок перечисления безразличен. В каждый конкретный момент времени может быть открыто не более одного массива из находящихся на одной магнитной ленте.

В целом структура раздела оборудования следующая:

РАЗДЕЛ ОБОРУДОВАНИЯ.
[СЕКЦИЯ КОНФИГУРАЦИИ.
[ИСХОДНАЯ-МАШИНА. МИНСК-32.]
РАБОЧАЯ-МАШИНА. МИНСК-32 [; РАЗМЕР ПАМЯТИ целое ЛИСТОВ].
[СПЕЦИАЛЬНЫЕ-НАЗВАНИЯ. название-устройства-1 ЕСТЬ условное-название-1. [название-устройства-2 ЕСТЬ условное-название-2. ...]
[СЕКЦИЯ ВВОДА-ВЫВОДА.
УПРАВЛЕНИЕ-МАССИВАМИ.
ДЛЯ название-массива ПРЕДНАЗНАЧИТЬ название-устройства-3 [, название-устройства-4]
[; СО СМЕНОЙ КАТУШЕК] [РЕЗЕРВИРОВАТЬ целое ДОПОЛНИТЕЛЬНЫХ ЗОН].
[УПРАВЛЕНИЕ-ВВОДОМ-ВЫВОДОМ.
[ПЕРЕПРОГОН [НА ЛЫ] ПОСЛЕ
{КОНЦА КАТУШКИ / целое ЗАПИСЕЙ} название-массива-1 ...]
[ОБЩАЯ ЗОНА ДЛЯ название-массива-2, название-массива 3 [, название-массива-4 ...] ...]
[НА ОДНОЙ КАТУШКЕ название-массива-5, название-массива-6 [, название-массива-7 ...] ...]]

Пример раздела оборудования:

РАЗДЕЛ ОБОРУДОВАНИЯ.
СЕКЦИЯ КОНФИГУРАЦИИ.
ИСХОДНАЯ-МАШИНА. МИНСК-32.
РАБОЧАЯ-МАШИНА. МИНСК-32; РАЗМЕР ПАМЯТИ 32 ЛИСТОВ.
СПЕЦИАЛЬНЫЕ-НАЗВАНИЯ.
ПЧ 1 ЕСТЬ ПЕЧАТЬ.
СЕКЦИЯ ВВОДА-ВЫВОДА.
УПРАВЛЕНИЕ-МАССИВАМИ.
ДЛЯ СПРАВОЧНЫЙ-1 ПРЕДНАЗНАЧИТЬ МЛ 1; СО СМЕНОЙ КАТУШЕК.
ДЛЯ СПРАВОЧНЫЙ-2 ПРЕДНАЗНАЧИТЬ МЛ 2.

ДЛЯ ПРОМЕЖУТОЧНЫЙ ПРЕДНАЗНАЧИТЬ МЛ3.
УПРАВЛЕНИЕ-ВВОДОМ-ВЫВОДОМ. ОБЩАЯ ЗОНА ДЛЯ СПРАВОЧНЫЙ-2, ПРОМЕЖУТОЧНЫЙ
ПЕРЕПРОГОН ПОСЛЕ 100 ЗАПИСЕЙ СПРАВОЧНЫЙ-1.

4.4. РАЗДЕЛ ДАННЫХ

В КОБОЛе все вводимые и выводимые данные представляют собой массивы. В разделе данных описывается внутренняя структура всех массивов. Под внутренней структурой понимается структура всех записей, входящих в соответствующие массивы, количество записей в физической зоне массива, наличие меток, метод организации массива и т. п. В этом разделе также определяются и назначаются рабочие области, дается описание промежуточных результатов и констант, определяемых пользователем, а также общих данных этой программы и внешних по отношению к ней других КОБОЛ-программ.

Как отмечалось ранее, под массивом понимается некоторая совокупность записей, в свою очередь записи представляют собой логически связанную совокупность реквизитов (данных).

Для того чтобы можно было ссылаться на данные в записи, в последней выделяются составные части, которые тоже можно подразделять, чтобы позволить более детальные ссылки к данным. В зависимости от того, подразделяются данные на составные части или нет, они называются групповыми или элементарными. Элементарное данное — это единица информации, которая не может быть логически делима. Элементарное данное будем называть реквизитом.

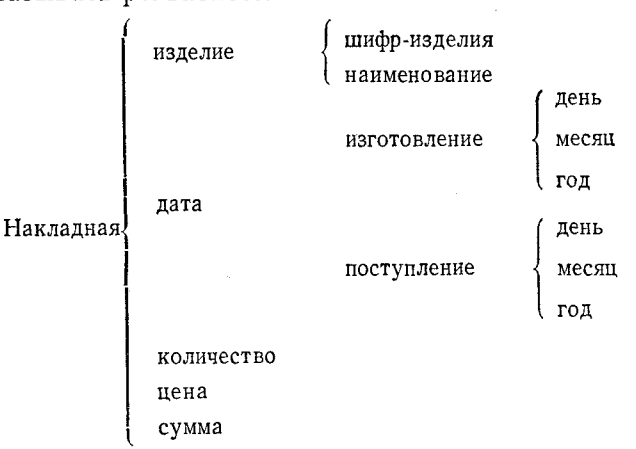


Рис. 4.2. Схема подчиненности данных в документе «Накладная»

Например, запись «Накладная» (рис. 4.2) можно разделить на пять данных: «изделие», «дата», «количество», «цена» и «сум-

ма», из которых «изделие» и «дата» являются групповыми, так как они разделены на более детальные данные. В этом примере данные «шифр-изделия», «наименование», «день», «месяц», «год», «количество», «цена», «сумма» являются реквизитами.

В общем случае групповое данное состоит из реквизитов или других групповых данных.

Общий вид раздела данных следующий:

РАЗДЕЛ ДАННЫХ.
[СЕКЦИЯ МАССИВОВ.
Статьи описания массивов.
Статьи описания записей.]
[СЕКЦИЯ РАБОЧЕЙ-ПАМЯТИ.
Статьи описания несвязанных рабочих полей или констант.
Статьи описания связанных рабочих полей.]
[СЕКЦИЯ СВЯЗИ.
Статьи описания несвязанных полей данных.
Статьи описания связанных полей данных.]

Описание входных и выходных массивов, промежуточных результатов и общих данных дается в разделе данных с помощью так называемых статей описания, состоящих в свою очередь из фраз описания. Каждая статья оканчивается точкой, фразы статьи отделяются точкой с запятой или пробелами.

Раздел данных, как видно из общего вида раздела, состоит из трех секций, которые называются СЕКЦИЯ МАССИВОВ, СЕКЦИЯ РАБОЧЕЙ-ПАМЯТИ и СЕКЦИЯ СВЯЗИ.

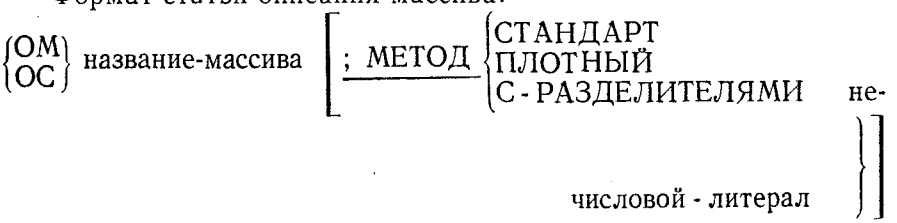
Секция массивов состоит из статей описания массивов и следующих за каждой из них статей описания записей, входящих в массивы. После описания первого массива должны следовать статьи описания всех записей, входящих в этот массив, затем — описание второго массива и соответственно всех записей, входящих в этот массив и т. д.

Секция рабочей памяти и секция связи состоят из статей описания несвязанных и связанных полей данных.

Любая из секций раздела данных не является обязательной. Если в данной программе какая-нибудь секция не нужна, ее заголовок не упоминается в разделе данных, заголовок же раздела обязателен и в том случае, если отсутствуют все секции раздела данных.

Секция массивов. Рассмотрим структуру секции массивов, состоящей из статей описания массивов и записей.

Формат статьи описания массива:



[; В БЛОКЕ целое ЗАПИСЕЙ] ; МЕТКИ { СТАНДАРТНЫ, ОПУЩЕНЫ } ШИФР МАССИВА нечисловой-литерал , ПЕРИОД ГОДНОСТИ целое-2] }

Если описание массива составляется для сортируемого массива, то эта статья должна начинаться с ОС, для всех остальных массивов — с ОМ. Отметим, что ОС и ОМ являются основными служебными словами КОБОЛа. Фраза МЕТОД { СТАНДАРТ ПЛОТНЫЙ С-РАЗДЕЛИТЕЛЯМИ нечисловой—литерал } используется для указания способа представления данных на носителе.

В КОБОЛе для ЭВМ «Минск-32» допускается обработка информации, подготовленной на перфолентах, перфокартах и магнитных лентах. Кроме того, массивы данных могут быть выведены на устройство печати. Каждому носителю поставлен в соответствие стандартный метод записи. Кроме того, на магнитной ленте может использоваться еще один метод — плотный.

Стандартный метод представления данных на перфоленте использует разделители. При этом стандартными разделителями являются символ «надчеркивание» в качестве разделителя данных и символ «ромбик» в качестве разделителя записей.

Во всех остальных случаях используется жесткий формат, означающий, что любое значение данного занимает фиксированное количество позиций, рассчитанное на максимально допустимый размер данного. При этом если число символов в значении данного меньше размера этого данного по описанию, то это значение дополняется до описанного размера в случае алфавитно-цифрового данного пробелами справа, а в случае цифрового — нулями слева и справа в зависимости от положения десятичной точки. При стандартном методе организации данных на магнитной ленте каждому реквизиту отводится одна или несколько ячеек, другими словами, информация распакована по ячейкам. При плотном методе значения реквизитов располагаются в ячейках подряд, при этом возможен перенос некоторых разрядов реквизита в другую ячейку. Однако каждая запись должна занимать целое количество ячеек. Если поля записи не занимают целое количество ячеек, оставшееся свободное место в последней ячейке заполняется нулями.

Вариант МЕТОД СТАНДАРТ используется, если массив представлен на носителе в стандартном методе. Вариант МЕТОД

ПЛОТНЫЙ может быть использован только для массива на магнитной ленте. Вариант МЕТОД С-РАЗДЕЛИТЕЛЯМИ позволяет определить нестандартные разделители для массивов на перфоленте. Нечисловой-литерал в формате должен состоять из двух любых литер, кроме кавычек и логического нет. Первая литера представляет собой разделитель данных, а вторая — разделитель записей. Кроме того, разделителем записей не могут быть литеры нуль (0) и звездочка (*). Фраза обязательна, если массив представлен на перфоленте и имеет нестандартные разделители или если для массива, расположенного на магнитной ленте, выбран плотный метод записи. В остальных случаях фраза может быть опущена.

Фраза с форматом В БЛОКЕ целое-1 ЗАПИСЕЙ используется для сообщения транслятору информации о размере физической зоны (блока) массива. Для каждого массива по указанному этой фразой размеру блока резервируется в оперативной памяти зона ввода-вывода, в которую вводится или из которой выводится на внешний носитель очередной блок массива. Если в массиве несколько типов записей, транслятор отведет место в памяти для указанного количества записей по максимальному размеру записи. Если в блоке, необходимом для соответсвующего массива, должно помещаться больше одной записи, то эта фраза обязательна. Кроме того, эта фраза обязательна, если массив на магнитных лентах представлен в плотном методе записи или если для массива на перфоленте используется фраза С-РАЗДЕЛИТЕЛЯМИ.

Если фраза отсутствует, размер блока принимается равным размеру наибольшей записи соответствующего массива. Размер блока равен размеру максимальной записи, умноженной на число записей в блоке.

При блочной организации информационных массивов с магнитной ленты можно прочитать только такую зону (блок), какая была записана.

Вопрос об оптимальном размере блока массива является вопросом большой важности, поскольку от этого зависит частота обращения к носителю массива информации, а значит и время работы программы. Ниже описан метод определения оптимального размера блока.

Ставится следующая задача. Для некоторой подсистемы (или системы), реализующей обработку информации, программа разбивается на отдельные этапы, этапы между собой связаны информационно и имеют некоторый приоритет выполнения, основанный главным образом на информационной преемственности. В каждом из этапов используются некоторые массивы информации, хранящиеся на МЛ, при этом эти массивы могут получаться в результате работы соответствующего этапа или использоваться для получения новых массивов. На каждом из этапов часть оперативной памяти будет занята программой, резидент-

ной частью операционной системы, оставшаяся свободная часть оперативной памяти может быть использована для сегментов, участвующих в работе этапа массивов.

Очевидно, что, если массив используется в программе всей подсистемы как готовый (он должен быть получен в другой подсистеме), можно условиться, что свободная оперативная память этапов, в которых массив используется, должна быть уменьшена на длину блока этого массива. Необходимо для всех массивов, получаемых и используемых в программе подсистемы, найти длину блока (сегмента) с условием, что суммарное количество обращений к магнитной ленте было минимальным.

Формализованная постановка задачи заключается в следующем. Дан функционал

$$T(x_1,\ x_2,\ \ldots\ x_n) = f(X) = \sum_{i=1}^{n} \alpha_i \frac{M}{x_i} \tag{4.1}$$

при условиях:

$$\left.\begin{aligned} &\sum_{i \in p_1} x_i \leqslant A_1;\\ &\sum_{i \in p_2} x_i \leqslant A_2;\\ &\sum_{i \in p_k} x_i \leqslant A_k; \end{aligned}\right\} \tag{4.2}$$

$$\left.\begin{aligned} &A_j > 0,\ j = 1,\ 2,\ \ldots,\ k;\\ &x_i \geqslant 1,\ i = 1,\ 2,\ \ldots,\ n;\\ &x_i \text{ — целые;} \end{aligned}\right\} \tag{4.3}$$

x_i — длина сегмента i-го массива. Длину массива и его сегмента будем измерять в некоторых условных единицах, например в символах или словах;

M_i — длина i-го массива. Всего в программе подсистемы используется n массивов. Отметим, что используемые массивы из других подсистем в это число не включаются;

α_i — количество использований (записей или чтений) i-го массива. Очевидно, что $\alpha_i \geqslant 1$;

k — количество этапов в программе подсистемы;

A_j — длина оперативной памяти на j-м этапе, которую можно использовать в качестве памяти для сегментов массивов.

Необходимо найти такие $x_1,\ x_2,\ \ldots,\ x_n$, чтобы функционая (4.1) принимал минимальное значение. Приведем алгоритм решения сформулированной задачи.

1. Выберем начальное значение $X^0 = (x_1^0,\ x_2^0,\ \ldots,\ x_n^0)$. Очевидно, что функционал (4.1) является выпуклой вниз функцией и на границе принимает максимальные значения. В качестве на-

чальных значений можно выбрать $x_i^0 = 1$ ($i = 1, 2, \ldots, n$). В случае если рассматриваемые массивы представляют собой массивы с постоянной длиной записи, то в качестве начальных значений можно выбрать $h_1, h_2, \ldots, h_n$, где h_i — длина записи i-го массива ($i = 1,2, \ldots, n$).

2. Рассчитываем значение функционала (4.1) в выбранной точке X^0:

$$T_0 = f(X^0).$$

3. Рассматриваем множество точек:

$$X_i = \{x_1^0, \ldots, x_{i-1}^0, x_i^0 + h_i, \ldots, x_n^0\},$$

где h_i — длина записи i-го массива или единица.

4. Из множества точек X_i ($i = 1, 2, \ldots, n$) строим подмножество точек

$$N = \{X_j, f(X_j) < f(X^0)\},$$

удовлетворяющее ограничениям (4.2).

5. Если множество N пусто, то точка X^0 является искомым решением.

6. Если множество N не пусто, то выбираем точку X_{i_0} таким образом, чтобы

$$f(X_{i_0}) = \min_{j \in N} f(x_j).$$

7. Полученную точку X_{i_0} рассматриваем в качестве исходной точки X^0 для этапа 2 и повторяем этапы 2—7.

На основании ранее приведенных характеристик функционала (4.1) и ограничений (4.2) и (4.3) очевидно, что найденная после описанного направленного перебора точка будет искомым решением.

Фраза с форматом

МЕТКИ { СТАНДАРТНЫ, ШИФР МАССИВА нечисловой-литерал, ПЕРИОД ГОДНОСТИ целое-2] ОПУЩЕНЫ }

используется для определения начальной и конечной меток массива. Начальная метка является частью начального контрольного блока массива и содержит наименование (шифр) массива, дату образования и срок годности массива. Конечная метка массива является частью конечного контрольного блока и содержит наименование массива.

Нечисловой литерал в формате не должен содержать более 5 литер. Этот литерал представляет собой имя массива. При проверке начальной метки входного массива нечисловой литерал, указанный во фразе, будет сравниваться с соответствующим именем массива, поставленного на устройство. Для выходного

массива указанный нечисловой литерал будет использоваться при создании метки. Значение шифра массива используется также в сообщениях программ ДИСПЕТЧЕРа для указания массива, к которому относится сообщение. Имя массива в отличие от шифра не используется при оформлении массива на внешнем носителе, а служит лишь для называния массива в программе.

Фраза ПЕРИОД ГОДНОСТИ используется для определения срока годности информации данного массива для данной задачи. Для выводимого массива формируется срок годности и записывается в начальном контрольном блоке. Целое-2 не должно быть больше 999.

Для входных массивов ПЕРИОД ГОДНОСТИ не используется. Если ПЕРИОД ГОДНОСТИ не указан, то он считается равным нулю. Если использована фраза МЕТКИ ОПУЩЕНЫ, предполагается, что шифр массива и период годности нулевые. Для выходных массивов на магнитной ленте не рекомендуется использовать фразу МЕТКИ ОПУЩЕНЫ. Как правило, эта фраза применяется к массивам, выводимым на печать. Наряду со стандартными метками массива, содержащими шифр массива и период годности, пользователь может вводить для опознания и контроля информации свои метки, описывая их в разделе данных как отдельный тип записи. В этом случае в разделе процедур должна быть организована обработка соответствующих записей — меток, как обычных записей.

Примеры статей описания массива:

ОМ УГЛОВЫЕ-СПЕЦИФИКАЦИИ; В БЛОКЕ 50 ЗАПИСЕЙ; МЕТКИ СТАНДАРТНЫ, ШИФР МАССИВА 'ГРАФ', ПЕРИОД ГОДНОСТИ 30.

ОС УЧЕТ-ДЕТАЛЕЙ; В БЛОКЕ 35 ЗАПИСЕЙ; МЕТКИ СТАНДАРТНЫ, ШИФР МАССИВА 'УД'.

Выше рассматривалось понятие записи. Посмотрим, как организованы данные внутри записи, на примере документа СПИСОК-СОТРУДНИКОВ, представленного по форме:

```
                                  ( ФАМИЛИЯ
                                  ( ИМЯ
                                  ( ОТЧЕСТВО
СПИСОК-СОТРУДНИКОВ ( ЛИЧНЫЕ-ДАННЫЕ ( ДАТА-РОЖДЕНИЯ ( МЕСЯЦ
                   ( ДОЛЖНОСТЬ     (              ( ГОД
                   ( ЗАРПЛАТА      (
```

В нашем примере СПИСОК-СОТРУДНИКОВ, ЛИЧНЫЕ-ДАННЫЕ и т. д.— названия данных. Каждое название в КОБОЛе должно представлять собой одно слово пользователя, поэтому внутри названия пробел запрещен. Для удобочитаемости названий используется дефис.

Все данные, составляющие документ, находятся в некотором отношении подчиненности и составляют иерархическую структу-

ру. Структуру данного документа можно представить в виде схемы (рис. 4.3). Здесь видна некоторая ступенчатость, т. е. одни названия располагаются на более высоком уровне, другие — на более низких. Так, названия ЛИЧНЫЕ-ДАННЫЕ, ДОЛЖНОСТЬ, ЗАРПЛАТА, именующие непосредственные составляющие записи, находятся на уровень ниже названия записи СПИСОК-СОТРУДНИКОВ. Названия ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО, ДАТА-РОЖДЕНИЯ также находятся на одном уровне, причем этот уровень ниже уровня, на котором расположены названия ЛИЧНЫЕ-ДАННЫЕ, ДОЛЖНОСТЬ, ЗАРПЛАТА.

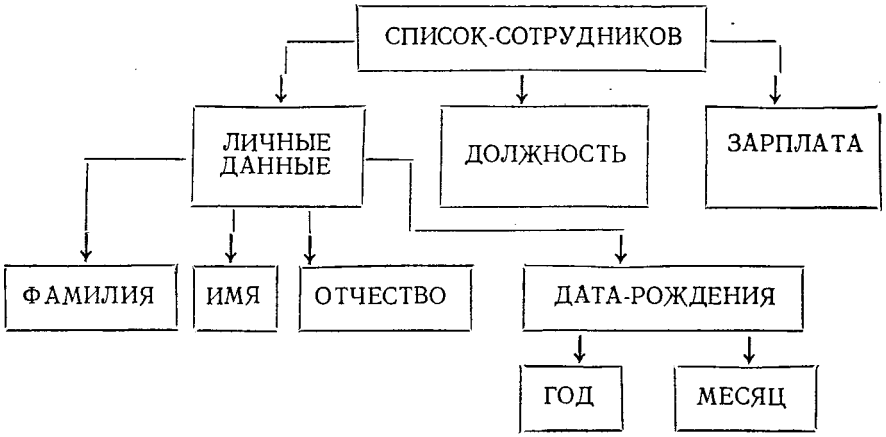


Рис. 4.3. Структура документа СПИСОК-СОТРУДНИКОВ

Иерархическая структура участвующих в обработке записей (документов) должна быть описана в разделе данных КОБОЛ-программы. Для того чтобы задать положение данных на ступенях иерархической лестницы, вводится понятие уровня.

Описание записи состоит из ряда статей описания данных, входящих в запись. Каждое данное в записи должно быть описано в отдельной статье. Порядок следования статей описания данных должен соответствовать расположению данных в документе, точнее — порядку их следования на перфоносителе. Из этого следует, что к составлению описаний данных можно приступить только после определения схемы перфорации соответствующего документа. Соподчиненность данных (иерархия данных) определяется номерами уровня. В качестве номеров уровня используются целые числа от 1 до 10.

Формат статьи описания данных:

номер-уровня { название-данного-1 [; ПЕРЕОПРЕДЕЛЯЕТ название-данного-2] / ЗАПОЛНИТЕЛЬ / ЗАП }

[; {ШАБЛОН / Ш} строка-литер] [; {ЗНАЧЕНИЕ / ЗНАЧ} {литерал / фигуральная-константа}]

[; ПОВТОРЯЕТСЯ целое {РАЗ / РАЗА}] [;ИНДЕКСИРУЕТСЯ название-индекса]

[;ДЛЯ {ВЫВОДА / ВЫЧИСЛЕНИЙ / ВЫЧ}] [;ПРОБЕЛ КОГДА НУЛЬ]

Порядок следования фраз в статье описания данного произвольный, однако статья должна начинаться с номера уровня.

Понятие уровня составляет основу аппарата языка КОБОЛ для описания сложных иерархических структур данных. Минимальный номер уровня 01 присваивается записи. Любое данное, входящее в состав некоторого другого данного, имеет соответственно численно больший номер уровня.

Следует обратить внимание на то, что понятие номера уровня связок, введенное для организации ввода экономической информации с перфоленты (разд. 2.1), несколько отличается от понятия уровня в КОБОЛе. Основное отличие состоит в том, что уровень связок устанавливает некоторый физический порядок следования реквизитов. Понятие уровня в КОБОЛе введено для отражения логической подчиненности групповых данных и реквизитов.

Пример.

Присвоим номера уровней данным документа «Накладная», структура которого приведена на рис. 4.2.

```
1 НАКЛАДНАЯ
    2 ИЗДЕЛИЕ
    3 ШИФР-ИЗДЕЛИЯ
    3 НАИМЕНОВАНИЕ
2 ДАТА
    3 ИЗГОТОВЛЕНИЕ
    4 ДЕНЬ
    4 МЕСЯЦ
    4 ГОД
3 ПОСТУПЛЕНИЕ
    6 ДЕНЬ1
    6 МЕСЯЦ1
    6 ГОД1
    2 КОЛИЧЕСТВО
    2 ЦЕНА
    2 СУММА
```

В этом описании номера уровней 1,2,3,4,6 определяют, что запись НАКЛАДНАЯ содержит в качестве подразделов следующие данные: ИЗДЕЛИЕ, ДАТА, КОЛИЧЕСТВО, ЦЕНА и СУММА. В свою очередь групповое данное ИЗДЕЛИЕ содержит ШИФР-ИЗДЕЛИЯ и НАИМЕНОВАНИЕ, данное ДАТА-ИЗГОТОВЛЕНИЕ и ПОСТУПЛЕНИЕ, групповые данные ИЗГОТОВЛЕНИЕ и ПОСТУПЛЕНИЕ содержат соответственно ДЕНЬ, МЕСЯЦ, ГОД.

Приведенное описание накладной соответствует документу следующей формы:

Накладная

Изделие		Дата						Количество	Цена	Сумма
		Изготовление			Получение					
Шифр изделия	Наименование	день	месяц	год	день	месяц	год			

Сформулируем правила, которыми следует руководствоваться при выборе номеров уровней.

1. Уровень 01 применяется только с названием записи.
2. Данные, входящие в одну группу и непосредственно составляющие групповое данное, должны иметь один и тот же номер уровня.
3. Номер уровня данного Б, содержащегося непосредственно в данном А, должен быть больше номера уровня данного А.
4. Номера уровней не обязательно должны образовывать последовательность чисел натурального ряда.
5. Номера уровней могут иметь значения от 1 до 10.
6. Номер уровня 77 зарезервирован для обозначения несвязанных рабочих полей в секции рабочей памяти.

Приведем формальное правило, определяющее непосредственную подчиненность данных. Данное Б непосредственно содержится в данном А, если выполняются следующие три условия: статья описания данного Б следует за статьей описания А; данное Б имеет номер уровня больший, чем А; между статьями описания А и Б нет статьи описания данного с номером уровня меньшим, чем у Б.

Выбранные (по сформулированным правилам) номера уровней в совокупности с порядком следования статей описания данных определяют как взаимосвязь данного с другими данными, так и порядок следования данных в документе и на соответствующем машинном носителе.

Рассмотрим назначение других фраз в статье описания данных. Любое данное (название-данного-1), на которое есть ссылка в разделе процедур, должно быть описано фразой с форматом:

номер-уровня название-данного-1 [ПЕРЕОПРЕДЕЛЯЕТ название-данного-2].

Фраза ПЕРЕОПРЕДЕЛЯЕТ служит для более эффективного использования оперативной памяти машины за счет того, что одно и то же рабочее поле будет использоваться для помещения в нем разных величин, которые, естественно, не должны храниться в поле одновременно. Название-данного-1 будем называть переопределяющим данным, название-данного-2 — переопределяемым. Эта фраза может использоваться для переопределения однотипных данных. Для этого необходимо, чтобы номера уров-

ней названия-данного-1 и названия-данного-2 были одинаковыми. В общем случае статьи описания переопределяющих и переопределяемых данных должны совпадать. Исключение составляет фраза ЗНАЧЕНИЕ, которая не должна использоваться при описании переопределяющего данного, но может быть использована при описании переопределяемого. Статьи переопределяющего данного должны непосредственно следовать за статьями переопределяемого. Не допускается иерархическое переопределение данных. Если, например, переопределяется некоторое групповое данное, то никакие групповые или элементарные данные внутри этого группового не должны иметь в описании фразу ПЕРЕОПРЕДЕЛЯЕТ.

В секции массивов не рекомендуется использовать фразу ПЕРЕОПРЕДЕЛЯЕТ на уровне 01, так как всем записям одного массива автоматически назначается одна и таже зона записи.

Пример.

```
2 ВЫРАБ-СДЕЛЬНАЯ.
  3 ЦЕНА-ИЗДЕЛИЯ ШАБЛОН 9Т99.
  3 КОЛИЧ-ИЗДЕЛИЙ ШАБЛОН 99.
2 ВЫРАБ-СТАВКА ПЕРЕОПРЕДЕЛЯЕТ ВЫРАБ-СДЕЛЬНАЯ.
  3 ЦЕНА-ЧАСА ШАБЛОН 9Т99.
  3 КОЛИЧ-ЧАСОВ ШАБЛОН 99.
2 ЗАРПЛАТА ... .
```

В данном примере переопределяется групповое данное ВЫРАБ-СДЕЛЬНАЯ.

В оперативной памяти данным ЦЕНА-ЧАСА и КОЛИЧ-ЧАСОВ будет отведено то же место, что и данным ЦЕНА-ИЗДЕЛИЯ и КОЛИЧ-ИЗДЕЛИЙ.

Фраза ЗАПОЛНИТЕЛЬ, или сокращенно ЗАП, употребляется в статье описания в том случае, если статья описывает реквизит, которому не надо присваивать имя, так как на него нет ссылок в разделе процедур, но для которого требуется отвести место в памяти.

Если массив состоит из записей разных типов, то для их идентификации используется шифр записи. Шифр должен быть одним из реквизитов записи. Для задания шифра записи используется фраза ЗАПОЛНИТЕЛЬ. Для каждого типа записи наличие этой фразы обязательно. При этом следует иметь в виду, что для описания шифра записи нельзя употреблять никакие названия данных, кроме фразы ЗАПОЛНИТЕЛЬ. Шифр записи должен быть задан нечисловым литералом и содержать не более трех литер. Шифры записей одного массива должны иметь одинаковое описание (содержать одинаковое количество литер в шаблоне). Значение шифра записи в каждом конкретном случае указывается во фразе ЗНАЧЕНИЕ. Кроме того, использование фразы ЗАПОЛНИТЕЛЬ в описании выходного массива позволяет управлять форматом по горизонтали при выдаче массива на печать. Например, интервал между графами некоторого выходного документа можно описать как ЗАПОЛНИТЕЛЬ, ука-

зав с помощью фраз ШАБЛОН и ЗНАЧЕНИЕ длину интервала и символы, которыми он должен быть заполнен.

Например, использование фразы ЗАПОЛНИТЕЛЬ в следующем описании записи выходного массива:

```
1 НАКЛАДНАЯ...
2 ШИФР-ИЗДЕЛИЯ...
2 ЗАП ШАБЛОН А(4) ЗНАЧЕНИЕ ПРОБЕЛ.
2 НАИМЕНОВАНИЕ...
```

обеспечивает то, что при выводе на печать между графами ШИФР-ИЗДЕЛИЯ и НАИМЕНОВАНИЕ будет интервал в четыре пробела.

Как уже указывалось, с помощью номеров уровней в КОБОЛе задается структура документа, определяющая, из каких составляющих он состоит, как эти составляющие подразделяются и т. д. Однако для того чтобы можно было обрабатывать документы, необходимо указать, из каких литер и в каком количестве составлено каждое данное документа, т. е. определить его класс и размер. Разумеется, имеет смысл говорить о классе и размере только для реквизита (элементарного данного), поскольку групповое данное может включать в качестве составляющих реквизиты различных классов.

Фраза ШАБЛОН служит для описания реквизитов. Слово ШАБЛОН можно сокращенно записывать как Ш. С помощью этой фразы приводятся характеристики реквизитов. Кроме того, ШАБЛОН может применяться для редактирования, с тем чтобы данные выдавались в удобочитаемом виде. Все реквизиты (и только они) должны быть описаны с помощью этой фразы. Строка-литер шаблона представляет собой некоторую комбинацию литер, при этом устанавливается позиционное соответствие между литерами шаблона и символами описываемого реквизита. Такое соответствие дает транслятору информацию о типе каждого символа в реквизите. К допустимым в шаблоне литерам относятся 9, А, Х, Т, М, З. Они несут следующую информацию:

9 — в соответствующей позиции данного будет цифра;

А — соответствующая позиция данного может быть занята буквой или пробелом;

Х — в соответствующей позиции данного может стоять любая литера из набора литер машины;

Т — положение неявной (подразумеваемой) десятичной точки внутри описываемого цифрового данного. Неявная десятичная точка не занимает позицию во внутреннем представлении данного (в зоне записи), однако она учитывается при определении его размера в зоне ввода-вывода;

М — место неявной десятичной точки вне поля, занимаемого данным. Литеры М в строке-литер шаблона могут стоять слева или справа. Если М слева, то предполагается, что неявная десятичная точка находится левее самого левого М, а если справа — то правее самого правого. М не учитываются при подсчете

размера данного во внутреннем представлении, но учитываются при выполнении арифметических операций и операций перемещения над таким данным. При подготовке данных, шаблон которых содержит М, соответствующие им нулевые позиции не перфорируются. Например, значение 30000 данного с шаблоном 99МММ перфорируется как 30;

З — значения цифрового данного могут быть как положительными, так и отрицательными. При отсутствии З предполагается, что данное всегда положительно. З должно стоять в шаблоне первым. При подсчете размера данного во внутреннем представлении З не учитывается.

При написании шаблона для сокращения вместо ряда одинаковых литер шаблона можно поставить только одну, за которой в скобках указать целым числом количество его повторений.

Примеры.

1. АААААА эквивалентно А(6).
2. 99999 эквивалентно 9(5).

В приведенных ниже примерах дан шаблон и допустимые значения соответствующих реквизитов.

1. 9(5)	34201
2. А(4)	ДАТА
	ВИД_
3. Х(3)	М1_
	Н-Д
4. 9Т9(2)	1.05
5. 9(3)ММ	24700

Для данного с таким шаблоном место в памяти отводится только для первых трех цифр.

6. М(4) 9(2)	000021

Для данного с таким шаблоном место в памяти отводится только для последних двух цифр.

7. 39(3)Т99	−147.56
	+395.12

Приводимые ниже литеры шаблона относятся к группе литер редактирования. Литеры редактирования разделяются на вставляемые и замещающие.

Вставляемые литеры:

. — явная десятичная точка. В отличие от неявной десятичной точки занимает в данном позицию и учитывается при определении размера величины во внутреннем представлении;

— — минус одиночный. Величина А отредактируется к виду:

а) с пробелом на месте знака, если $A \geqslant 0$;

б) с минусом на месте знака, если $A < 0$;

\+ — плюс одиночный. Величина А отредактируется к виду:

а) с плюсом на месте знака, если $A \geqslant 0$;

б) с минусом на месте знака, если $A < 0$;

В — вставка пробела.

Кроме того, для редактирования может использоваться любой нечисловой литерал. Он переносится из шаблона без всяких изменений в соответствующие части редактируемого данного.

Тем самым допускается возможность при помощи шаблона выводить наименования единиц измерения и т. п.

Кавычки, заключающие литерал, в поле результата не вносятся. Всего в данном может быть не более 127 литер (включая вставленные).

Примеры.

В приведенных ниже примерах даны значение данного, шаблон и соответствующее значение данного после редактирования.

Знаком /\ отмечается положение неявной десятичной точки.

1.	870	— 9(3)	_870
2.	—75/\3	+99.9	—75.3
3.	2975	99'руб.'ВВ99'коп.'	29 руб._ _75 коп.
4.	— 32	— В99	— _32

Замещающие литеры:

П — подавление нулей. В позициях данного, где помещены П, ведущие нули заменяются пробелами;

* — звездочка. В отмеченных звездочкой позициях ведущие нули заменяются звездочками;

— — минус плавающий. Если знак минус появляется в шаблоне подряд более одного раза, то он воспринимается транслятором как плавающий минус. Подобно П, он служит для подавления ведущих нулей. В позициях данного, где помещены минусы, ведущие нули заменяются пробелами, а в последней из подавляемых позиций ставится знак минус, если величина $A<0$, или пробел, если величина $A\geqslant 0$;

\+ — плюс плавающий. Аналогичен плавающему минусу, но если величина $A\geqslant 0$, то в последней из подавляемых позиций ставится не пробел, а знак +.

При использовании редактирования нужно выполнять следующие правила: а) шаблон редактируемого данного не может содержать литеры замещения различных типов; б) среди и после литер замещения в шаблоне могут быть литеры вставки или вставляемые литералы. Перед символами замещения * и П разрешается ставить одиночные + или —; в) литеры замещения могут располагаться в строке литер шаблона слева от десятичной точки или соответствовать всем цифровым позициям данного. Если все цифровые позиции данного, значение которого равно нулю, представлены в шаблоне литерами замещения, то любое редактирование игнорируется и на месте всех позиций данного ставятся пробелы.

Примеры.

В приведенных ниже примерах даны значение данного, шаблон и соответствующее значение данного после редактирования.

1.	00075	ПП999	_ _075
2.	00075	**9(3)	** 075
3.	—00075	+(3)9(2)	_ _ —075

Данные, шаблон которых описывается с помощью литер 9, М, Т, З, относятся к цифровому классу. При записи шаблона циф-

рового данного надо помнить, что цифр в таком данном должно быть не более 18 (цифры в шаблоне могут быть представлены литерами 9 и М).

Данные, шаблон которых описан с помощью литеры А, относятся к буквенному классу; с помощью литеры Х — к буквенно-цифровому. Шаблон цифровых редактируемых данных может содержать только литеры 9 и литеры редактирования, причем литер, представляющих цифры (9, П, *, плавающие + и −), должно быть не более 18.

Все остальные комбинации литер шаблона недопустимы.

Фраза ЗНАЧЕНИЕ в секции массивов применяется для указания начального значения, придаваемого специфическому данному ЗАПОЛНИТЕЛЬ.

{ ЗНАЧЕНИЕ / ЗНАЧ } { литерал / фигуральная-константа }

Если фраза используется для указания значения шифра записи, она должна быть задана нечисловым литералом, содержащим не более трех литер.

Если в статье описания данного имеются и фраза ШАБЛОН, и фраза ЗНАЧЕНИЕ, то они должны быть непротиворечивы. Если данное цифрового класса, то значение его может быть задано числовым литералом или фигуральной константой НУЛЬ. Если данное буквенного или буквенно-цифрового класса, то его значение задается нечисловым литералом или любой фигуральной константой (НУЛЬ, ПРОБЕЛ, КАВЫЧКА, ВСЕ).

Фраза ЗНАЧЕНИЕ не должна использоваться в статье описания данного, содержащей фразу ПОВТОРЯЕТСЯ или ПЕРЕОПРЕДЕЛЯЕТ, или в статье, на которую эти фразы распространяются.

Например, недопустимо следующее описание:

2 СТРОКА; ПОВТОРЯЕТСЯ 9 раз.
3 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(4); ЗНАЧЕНИЕ А5В6.

Пример.

```
1 СТРОКА-ВЕД.
2 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(5); ЗНАЧЕНИЕ ПРОБЕЛ.
2 ПОР-НОМЕР; ШАБЛОН ПП9.
2 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(6); ЗНАЧЕНИЕ ВСЕ'_
2 ТАБ-НОМЕР; ШАБЛОН 9(4).
2 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(6); ЗНАЧЕНИЕ '_____'.
2 НАЧИСЛЕНО; ШАБЛОН П(3)99.99.
```

Фраза ПОВТОРЯЕТСЯ избавляет программиста от повторения описаний однородных, идентично описываемых, подряд расположенных данных, представляющих, например, строки или столбцы документа.

Формат фразы:

ПОВТОРЯЕТСЯ целое { РАЗ / РАЗА } [; ИНДЕКСИРУЕТСЯ название-индекса-1]

Целое не должно превышать 1023.

Фраза может быть употреблена на уровне группового данного и указывает в этом случае, что описание применимо к повторяющейся группе данных. Количество статей описания элементарных данных в повторяющейся группе не должно быть больше 63, а для сортируемого массива — 32. Недопустимо использование фразы на уровне 01.

Фразы описания, которые использованы в описании пункта наряду с фразой ПОВТОРЯЕТСЯ распространяются на каждое повторение этого пункта.

Фраза ЗНАЧЕНИЕ не должна использоваться в описании пункта наряду с фразой ПОВТОРЯЕТСЯ или в статье, на которую распространяется фраза ПОВТОРЯЕТСЯ.

Фраза ПОВТОРЯЕТСЯ может быть использована в статье описания данного, входящего в группу данных, описанную с фразой ПОВТОРЯЕТСЯ.

Однако «глубина вложенности» не может превышать 2. Например, описание

2 ... ПОВТОРЯЕТСЯ...
3 ...
3 ... ПОВТОРЯЕТСЯ...
4 ... ПОВТОРЯЕТСЯ...

запрещается, так как здесь глубина вложенности равна 3.

Пример.

Пусть есть таблица с данными о количестве студентов 9 факультетов университета. Допустим, что в ней приведены цифры за 1950, 1960 и 1970 гг., т. е. она имеет вид:

Факультет	Количество студентов		
	1950 г.	1960 г.	1970 г.
1. Биологический	70	95	100
2. Географический	50	80	95
3. Исторический	60	90	90
	.	.	
	.	.	.
.		.	.
9. Физический	100	125	250

В разделе данных она может быть описана следующим образом:

1 ТАБЛИЦА.
2 СТРОКА-ТАБЛ; ПОВТОРЯЕТСЯ 9 РАЗ ИНДЕКСИРУЕТСЯ ИНД1.
3 ФАКУЛЬТЕТ ШАБЛОН Х(30).
3 КОЛИЧ-СТУДЕНТОВ ШАБЛОН 999; ДЛЯ ВЫЧИСЛЕНИЙ; ПОВТОРЯЕТСЯ 3 РАЗА ИНДЕКСИРУЕТСЯ ИНД2.

В этом примере описана двумерная матричная структура. С помощью внешнего ПОВТОРЯЕТСЯ указано количество однородных строк (9), а с помощью внутреннего — количество однородных столбцов (3).

Название данного, имеющее в описании фразу ПОВТОРЯЕТСЯ или принадлежащее к группе, имеющей в описании фразу ПОВТОРЯЕТСЯ, может употребляться в разделе процедур только с индексом.

Вариант ИНДЕКСИРУЕТСЯ название-индекса-1 в фразе ПОВТОРЯЕТСЯ указывает транслятору, что для ссылок на повторяющееся данное или его подразделения будет использоваться указанное название индекса.

Название-индекса-1 никак не связано с иерархией данных и не требует описания в разделе данных.

Одно и то же название индекса не должно использоваться при описании нескольких таблиц или списков. Иными словами, каждое название индекса должно быть связано только с одной таблицей или списком.

Индексы (один или два) записываются в разделе процедур за названием данного в скобках и могут отделяться запятой. Индекс может быть названием данного (в том числе СЧЕТЧИК), названием индекса или целым.

Формат индексированного данного:

название-данного-1 ({ название-данного-2 / целое-1 / название-индекса-1 } [, { название-данного-3 / целое-2 / название-индекса-2 }])

Название-данного-1 должно быть описано с фразой ПОВТОРЯЕТСЯ или входить в состав группы, описанной с ПОВТОРЯЕТСЯ. Если название-данного-1 входит только в одну группу с ПОВТОРЯЕТСЯ, требуется только один индекс. Если название-данного-1 входит в группу ПОВТОРЯЕТСЯ, которая в свою очередь входит в группу ПОВТОРЯЕТСЯ, требуется два индекса. Первым записывается индекс, который относится к внешнему ПОВТОРЯЕТСЯ, а вторым — к внутреннему.

Название-данного-2 (название-данного-3) должно быть названием цифрового данного, шаблон которого не содержит точки (Т) и знака (З). Кроме того, шаблон такого данного должен содержать столько литер, сколько цифр в целом, указанном во фразе ПОВТОРЯЕТСЯ. Текущее значение этого данного определяет, какое повторение данного-1 требуется.

Название-данного-2 (название-данного-3) не должно требовать индексирования, т. е. его описание не должно содержать фразу ПОВТОРЯЕТСЯ или подчиняться описанию с такой фразой.

Название индекса не должно использоваться в одной ссылке в сочетании с названием данного или целым.

При использовании названия данного и целого в качестве индекса во время выполнения рабочей программы будет производиться преобразование этого данного в индексную форму, требуемую для определения машинного адреса. Использование

названия индекса дает возможность повысить эффективность рабочей программы.

Ниже приведены примеры использования индексов для ссылок на элементы таблицы, описанной в предыдущем примере.

КОЛИЧ-СТУДЕНТОВ(1, 2) — означает ссылку на количество студентов биологического факультета в 1960 г., равное 95;

КОЛИЧ-СТУДЕНТОВ(ИНД1, ИНД2) — означает ссылку на количество студентов физического факультета в 1970 г., равное 250, если значение ИНД1 было установлено на 9, а ИНД2 — на 3.

Фраза ДЛЯ { ВЫВОДА / ВЫЧИСЛЕНИЙ / ВЫЧ } служит для указания основного использования данного.

Фраза в описании записи может быть использована на любом уровне. Если она стоит на уровне группового данного, то ее действие распространяется на все элементарные данные, принадлежащие этому групповому.

Если в описании данного использован вариант ДЛЯ ВЫЧИСЛЕНИЙ, или сокращенно ВЫЧ, то предполагается, что оно будет использоваться преимущественно в вычислениях, например, в качестве аргумента арифметических операций. Такое данное будем называть вычислительным. Фраза может быть использована, следовательно, только для цифровых данных; во всех остальных случаях транслятором выдается сообщение об ошибке.

Вариант ДЛЯ ВЫВОДА употребляется в случае, если данное будет использоваться преимущественно для вывода. Имеет смысл использовать эту фразу только для цифровых данных. Фраза игнорируется, если она относится к буквенному, буквенно-цифровому или цифровому редактируемому данному.

Если ни одна из фраз не употреблена в описании данного, предполагается использование ДЛЯ ВЫВОДА.

Пример.

```
02 ФАМИЛИЯ; ШАБЛОН А(12)
02 ВЫРАБОТКА-СДЕЛЬНАЯ; ДЛЯ ВЫЧИСЛЕНИЙ.
   03 ЦЕНА-ИЗДЕЛИЯ; ШАБЛОН 9Т99.
   03 КОЛИЧЕСТВО-ИЗДЕЛИЯ; ШАБЛОН 9999.
```

В данном случае действие фразы ДЛЯ ВЫЧИСЛЕНИЙ, использованной на уровне группового данного, распространяется на элементарные данные «цена-изделия» и «количество-изделия», входящие в этот групповой.

Фраза ПРОБЕЛ КОГДА НУЛЬ используется для редактирования цифровых данных при выводе и означает замену данного пробелами, если его значение равно нулю.

Фраза может использоваться только на уровне реквизита.

Если фраза использована в описании буквенного или буквенно-цифрового данного или употреблена в описании данного наряду с фразой ДЛЯ ВЫЧИСЛЕНИЙ, транслятор выдает сообщение об ошибке.

Секция рабочей памяти. Секция рабочей памяти используется для описания полей, в которых во время выполнения рабочей программы хранятся промежуточные результаты или именуемые константы.

Вообще поля, описываемые в секции рабочей памяти, могут быть двух типов: несвязанные и связанные.

Несвязанные — это одиночные поля, которые не имеют подразделений и сами не являются подразделениями других полей. Для описания таких полей используется специальный номер уровня 77.

Связанные поля могут состоять из одной и более величин, сгруппированных для образования записи. Формат описания записи в секции рабочей памяти идентичен формату описания записи в секции массивов.

Структура секции рабочей памяти следующая:

СЕКЦИЯ РАБОЧЕЙ-ПАМЯТИ.
Статьи описания несвязанных полей.
Статьи описания связанных полей.

Сначала должны быть описаны все требуемые несвязанные поля, затем — связанные.

В описании данного могут быть использованы те же фразы, что и в секции массивов, со следующими ниже отличиями.

При использовании фразы ПЕРЕОПРЕДЕЛЯЕТ должны соблюдаться те же правила, что и в секции массивов, со следующими поправками: допускается переопределение на уровне записей и размер в ячейках переопределяющего поля может отличаться от размера переопределяемого поля, но не должен превосходить его.

Пример переопределения в секции рабочей памяти:

```
1 ТОВАРНЫЙ-СКЛАД.
2 НАИМЕНОВ-ПРОДУКЦИИ ШАБЛОН А(10).
2 КОЛИЧ-ИЗДЕЛИЙ ШАБЛОН 9(7).
2 ОТГРУЖЕНО-ЗА-ДЕНЬ ШАБЛОН 9(5).
1 ПРЕДПРИЯТИЕ ПЕРЕОПРЕДЕЛЯЕТ ТОВАРНЫЙ-СКЛАД.
2 ШИФР-ПРЕДПРИЯТИЯ ШАБЛОН Х(3).
2 ИНФОРМАЦИЯ-О-ПЛАНЕ ШАБЛОН 9(4).
```

Фраза ЗНАЧЕНИЕ в секции рабочей памяти используется для задания начальных значений рабочих полей, именуемых констант и заполнителей.

Фраза может быть использована как на уровне элементарного данного, так и на уровне группового. Если фраза употреблена на уровне группового данного, значение должно быть задано фигуральной константой или нечисловым литералом и поле, отведенное для этого данного, заполняется без рассмотрения свойств отдельных (групповых или элементарных) данных, входящих в это групповое.

Размер группового данного, значение которого задано нечисловым литералом, а также фигуральными константами — ПРОБЕЛ, КАВЫЧКА, ВСЕ нечисловой-литерал — не должен превышать 128 символов.

Гнездовое использование фразы ЗНАЧЕНИЕ недопустимо.

Примеры.

02 ВЕД-ЗАРП ШАБЛОН Х(18) ЗНАЧЕНИЕ 'ВЕДОМОСТЬ_ЗАРПЛАТЫ'. Литерал 'ВЕДОМОСТЬ_ЗАРПЛАТЫ' будет помещен в поле ВЕД-ЗАРП.

03 ЗАПОЛНИТЕЛЬ ШАБЛОН А(3) ЗНАЧЕНИЕ ПРОБЕЛ.

В рабочей программе будет обеспечена засылка трех пробелов в поле, отведенное под данный ЗАПОЛНИТЕЛЬ.

03 ЗАРПЛ ШАБЛОН 9(3)Т9(2) ЗНАЧЕНИЕ 120.62.

В поле данного ЗАРПЛ будет занесено 12062, но при обращении к данному это число будет интерпретироваться как 120.62.

77 ЧЕРТА ШАБЛОН Х(120) ЗНАЧЕНИЕ ВСЕ '—'.

Здесь задана «длинная» константа, состоящая из 120 минусов.

Фраза ПОВТОРЯЕТСЯ не может быть использована на уровне 01 и 77.

Фраза об использовании имеет следующий формат:

$$\text{ДЛЯ} \left\{ \begin{array}{l} \text{ВЫВОДА} \\ \text{ВЫЧИСЛЕНИЙ} \\ \text{ВЫЧ} \\ \text{ИНДЕКСА} \end{array} \right\}$$

Фразы ДЛЯ ВЫВОДА и ДЛЯ ВЫЧИСЛЕНИЙ (ВЫЧ) имеют такое же использование, как и в секции массивов.

Элементарное данное, описанное с фразой ДЛЯ ИНДЕКСА, называется индексом данного и используется для запоминания значения названия-индекса, употребленного во фразе ПОВТОРЯЕТСЯ. Значение соответствует номеру появления элемента таблицы (списка) и запоминается без какого-либо преобразования.

Если фраза ДЛЯ ИНДЕКСА появилась на уровне группового данного, то каждое элементарное данное этого группового является индексом. Само групповое данное не является индексом и не может быть использовано в операторе УСТАНОВИТЬ или в проверке отношения. Не разрешается использовать наряду с фразой ДЛЯ ИНДЕКСА другие фразы описания данного.

Секция связи. Секция связи служит для описания данных, определенных в некоторой внешней программе и передаваемых данной программе.

Данные, описанные в секции связи, не вызывают распределения памяти для них, поэтому они обязательно должны иметь соответствующее описание в секциях массивов или рабочей памяти каких-то других программ.

Структура секции связи аналогична структуре секции рабочей памяти и имеет вид:

СЕКЦИЯ СВЯЗИ.
Статьи описания несвязанных полей.
Статьи описания связанных полей.

В описании данного могут быть использованы те же фразы, что и в секции рабочей памяти.

Надо заметить, что хотя информация, находящаяся в зонах записей, доступна нескольким программам, т. е. записи одного массива могут обрабатываться несколькими программами, ввод-вывод этого массива не может быть поделен между программами, т. е. описание массива и операторы типа ОТКРЫТЬ, ЧИТАТЬ, ПИСАТЬ, ЗАКРЫТЬ для этого массива могут встретиться только в одной из групп связанных программ.

Пример раздела данных.

```
РАЗДЕЛ ДАННЫХ.
СЕКЦИЯ МАССИВОВ.
ОМ ВЫРАБОТКИ; В БЛОКЕ 50 ЗАПИСЕЙ;
МЕТКИ СТАНДАРТНЫ, ШИФР МАССИВА 'ВЫРАБ',
ПЕРИОД ГОДНОСТИ 200.
1 ВЫРАБОТКА-РАБОЧЕГО.
  2 ФИО; ШАБЛОН А(12).
  2 НАЗВАНИЕ-РАБОТЫ; ШАБЛОН Х(5).
  2 ВЫРАБОТКА; ШАБЛОН 9(4); ДЛЯ ВЫЧИСЛЕНИЙ.
ОМ РАСЦЕНКИ; МЕТКИ СТАНДАРТНЫ, ШИФР 'РАСЦЕ'.
1 ТАБЛИЦА-РАСЦЕНОК.
  2 РАСЦЕНКА; ПОВТОРЯЕТСЯ 20 РАЗ; ИНДЕКСИРУЕТСЯ ИНД.
    3 НАЗВ-РАБ; ШАБЛОН Х(5).
    3 ЦЕНА-ИЗДЕЛИЯ; ШАБЛОН 9(4); ДЛЯ ВЫЧИСЛЕНИЙ.
ОМ ПЛАТЕЖНАЯ-ВЕДОМОСТЬ; МЕТОД ПЛОТНЫЙ; В БЛОКЕ 20 ЗАПИСЕЙ;
    МЕТКИ СТАНДАРТНЫ, ШИФР 'ПЛВЕД'.
1 ПЛАТЕЖ-ВЕД.
  2 ФАМИЛИЯ; ШАБЛОН А(12).
  2 ЗАРПЛАТА; ШАБЛОН 9(6); ДЛЯ ВЫЧИСЛЕНИЙ.
СЕКЦИЯ РАБОЧЕЙ-ПАМЯТИ.
77 М; ШАБЛОН 99; ДЛЯ ВЫЧИСЛЕНИЙ.
77 ИНД; ДЛЯ ИНДЕКСА.
1 СТР-1.
  2 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(85); ЗНАЧЕНИЕ ВСЕ '—'.
1 СТР-2.
  2 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(30); ЗНАЧЕНИЕ ПРОБЕЛ.
  2 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(19); ЗНАЧЕНИЕ 'ПЛАТЕЖНАЯ ВЕДОМОСТЬ'.
1 СТР-3.
  2 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(10); ЗНАЧЕНИЕ ПРОБЕЛ.
  2 ФАМ; ШАБЛОН Х(12).
  2 ЗАПОЛНИТЕЛЬ; ШАБЛОН Х(4); ЗНАЧЕНИЕ '____'.
  2 ЗАРП; ШАБЛОН ППП9'_'РУБ'_'99'_КОП_'.
```

4.5. РАЗДЕЛ ПРОЦЕДУР

В разделе процедур исходной программы описываются действия, производимые над данными в процессе их обработки: перемещение данных с одного внешнего носителя на другой, разного рода вычисления, редактирование данных и т. п.

Возможности языка КОБОЛ позволяют выполнять сравнительно простые и более сложные операции. Примером первых могут служить операции вычисления, перемещения элементарных или групповых данных из одного поля памяти в другое, изменение последовательности выполнения операций. Более сложные операции выполняются для упорядочения массивов, ввода и вывода на внешние носители групп данных, однократного и многократного выполнения групп более простых операций, для вызова других программ и их выполнения.

Раздел процедур начинается с заголовка.

РАЗДЕЛ ПРОЦЕДУР [ИСПОЛЬЗУЯ [нечисловой-литерал-1, название-данного-1 [, название-данного-2...] [, нечисловой-литерал-2, название-данного-3 [, название-данного-4 ...] ...]].

Фраза ИСПОЛЬЗУЯ должна присутствовать в заголовке раздела процедур вызываемой программы, если она имеет с программой, которая ее вызывает, общие данные, описанные в секции связи.

Каждый нечисловой литерал представляет название общей области программы, полученной после трансляции. Литерал должен состоять не более чем из 5 литер.

Группирование данных в общие области производится по соображениям удобства обработки и экономичности описания их в нескольких программах. Например, данные Д1, Д2, Д3, Д4, Д5, Д6, используемые в вызываемых программах П1, П2 и П3, можно разбить на области ОБЩ1, ОБЩ2, ОБЩ3 следующего состава:

ОБЩ1 — Д1, Д2, Д3; ОБЩ2 — Д4, Д5; ОБЩ3 — Д6.

Если П1 использует вместе с вызывающей программой данные Д1, Д2, Д3, Д6, программа П2 — данные Д4, Д5, Д6, а программа П3 — данное Д6, то название раздела процедур будет иметь вид:

для П1

РАЗДЕЛ ПРОЦЕДУР ИСПОЛЬЗУЯ 'ОБЩ1', Д1, Д2, Д3, 'ОБЩ3', Д6.

для П2

РАЗДЕЛ ПРОЦЕДУР ИСПОЛЬЗУЯ 'ОБЩ2', Д4, Д5, 'ОБЩ3', Д6.

для П3

РАЗДЕЛ ПРОЦЕДУР ИСПОЛЬЗУЯ 'ОБЩ3', Д6.

Названия данных должны быть описаны в секции связи вызываемой программы на уровнях 01 и 77.

Список данных, следующих за нечисловым литералом, определяет размер и относительное расположение данных в общей области с названием, определенным нечисловым литералом, и должен соответствовать списку данных в секции связи вызываемой программы и секции массивов или секции рабочей памяти вызывающей программы. Названия общих областей в разделах данных соответствующих программ не указываются.

Вариант ИСПОЛЬЗУЯ название-данного-1 [название-данного-2, ...] в заголовке раздела процедур внутренней программы используется в случае, если внешняя программа передает параметры без названий общих областей. Количество, порядок и описания параметров обращения во внешней программе должны совпадать с количеством, порядком и описанием параметров в варианте

ИСПОЛЬЗУЯ нечисловой-литерал-1, название-данного-1 [, название-данного-2 ...] [, нечисловой-литерал-2, название-данного-3 [, название-данного-4 ...] ...].

Основу раздела процедур составляют глаголы, определяющие действия, которые необходимо выполнить.

Особое место в разделе процедур занимает ПРИМЕЧАНИЕ, с помощью которого можно ввести в программу необходимые комментарии. После слова ПРИМЕЧАНИЕ ставится точка, за которой может следовать комментирующее предложение — любая комбинация слов языка, ограниченная точкой.

Комбинация глагола со служебными словами и словами пользователя, допустимыми форматом глагола, образуют оператор (высказывание) языка. Раздел процедур полностью состоит из операторов. Синтаксис языка КОБОЛ позволяет строить из операторов синтаксические единицы разной степени сложности: предложения, параграфы, секции. Каждая из этих единиц в свою очередь может быть объектом действия некоторых глаголов языка.

Таким образом, предложение, параграф, секция являются как бы разными уровнями описания процедур программы.

Предложение состоит из одного или нескольких операторов. В конце его обязательно ставится точка. При объединении операторов в предложения следует иметь в виду, что для некоторых (логических) операторов выбор места точки имеет большое значение, поскольку граница предложения является для них сигналом к выбору альтернативного пути продолжения работы программы. В остальных случаях выбор места точки произволен.

Параграф образуется из одного или нескольких предложений. Ему должно быть присвоено название, которое ставится перед первым предложением параграфа и заканчивается точкой. В параграф входят все предложения до следующего названия параграфа.

Один или несколько параграфов можно объединить в секцию — самую крупную синтаксическую единицу языка. Каждая секция также должна быть названа. Для этого перед названием первого параграфа секции записывается слово СЕКЦИЯ, за которым (через пробел) следует название секции, заканчивающееся точкой. В секцию входят все параграфы до следующего названия секции (концом последней секции является конец программы). Поэтому если некоторые параграфы раздела процедур

необходимо объединить в секцию, то и всю последующую часть программы надо представить в виде секций.

Процедуры выполняются в той последовательности, в которой они записаны, до тех пор, пока не встретится глагол из группы глаголов изменения последовательности процедур. Последние позволяют изменять последовательность выполнения процедур или выполнять их циклически. При этом если переход осуществляется по названию параграфа, то начинает выполняться первый оператор этого параграфа, если же по названию секции — первый оператор первого параграфа секции.

В разделе процедур могут использоваться два типа операторов: повелительные и условные. Соответственно и предложения могут быть двух типов — повелительные и условные.

Оператор, который предполагает выбор одного действия из нескольких возможных в зависимости от истинности некоторого условия, называется условным. К условным операторам относятся операторы с глаголами ЕСЛИ, ВЫЧИСЛИТЬ (с фразой ПРИ ПЕРЕПОЛНЕНИИ), ЧИТАТЬ.

Условное предложение в общем случае состоит из одного или нескольких повелительных операторов, за которыми следует условный оператор, оканчивающийся точкой. Наличие повелительного оператора перед условным в условном предложении не обязательно. Однако сам условный оператор должен содержать в своем составе один или несколько повелительных операторов, что следует из форматов глаголов ЕСЛИ, ВЫЧИСЛИТЬ (с фразой ПРИ ПЕРЕПОЛНЕНИИ) и ЧИТАТЬ. Допустим только один условный оператор в условном предложении.

Например, возможно следующее условное предложение: ВЫЧИСЛИТЬ ДАННОЕ2=ДАННОЕ2 * ДАННОЕ1, ЕСЛИ ДАННОЕ2=0 ЧИТАТЬ МАССИВ-РЕЗУЛЬТАТ В КОНЦЕ ВЫПОЛНИТЬ БЛОК1, ЗАКРЫТЬ МАССИВ-НОВЫЙ, ОСТАНОВИТЬ РАБОТУ.

Оператор, который однозначно предписывает определенное действие, называется повелительным.

Для образования повелительных операторов используются следующие глаголы: ВОЙТИ, ВЫДАТЬ, ВЫЗВАТЬ, ВЫПОЛНИТЬ, ВЫЧИСЛИТЬ (без фразы ПРИ ПЕРЕПОЛНЕНИИ), ЗАКРЫТЬ, ИЗМЕНИТЬ, ОСТАНОВИТЬ, ОСВОБОДИТЬ, ОТКРЫТЬ, ПЕРЕЙТИ, ПИСАТЬ, ПОМЕСТИТЬ, ПРИНЯТЬ, ПРОСМОТРЕТЬ, СОРТИРОВАТЬ, УСТАНОВИТЬ.

Повелительное предложение состоит из одного или нескольких повелительных операторов и ограничено точкой. Причем если предложение содержит оператор с глаголом ПЕРЕЙТИ или ОСТАНОВИТЬ, то такой оператор должен быть последним в предложении.

Это правило справедливо и для группы повелительных операторов в форматах глаголов ЕСЛИ, ВЫЧИСЛИТЬ (с фразой ПРИ ПЕРЕПОЛНЕНИИ) и ЧИТАТЬ.

Арифметический оператор ВЫЧИСЛИТЬ. Оператор ВЫЧИСЛИТЬ служит для присвоения данному значения другого данного, литерала или арифметического выражения.

Формат оператора:

ВЫЧИСЛИТЬ название-данного = { название-данного-2 | числовой-литерал-1 | арифметическое-выражение }

[ОКРУГЛЯЯ] [ПРИ ПЕРЕПОЛНЕНИИ повелительный-оператор]

Арифметическое выражение — это комбинация названий данных, числовых литералов, СЧЕТЧИКа, соединенных знаками арифметических операций. Полный список знаков арифметических операций, определяющих соответственно действия сложения, вычитания, умножения, деления и возведения в степень следующий: +, —, *, /, **.). Порядок выполнения операций в арифметическом выражении определяется, как обычно, круглыми скобками и старшинством операций.

Все данные в арифметическом выражении должны представлять собой данные цифрового класса.

Показатель степени в операции возведения в степень должен быть целым числом, не превышающим 59. Если он задан нечисловым литералом, а названием данного, то как сам показатель, так и основание степени не должны содержать точки в шаблоне описания данного.

В операнде арифметической операции может быть не более 18 цифр, включая целую и дробную части. Промежуточный результат вычислений тоже должен содержать не более 18 цифр, в противном случае будут потеряны значащие цифры. Необходимо иметь в виду, что количество цифр и место точки (т. е. количество цифр в дробной части числа) для результата арифметической операции определяются шаблоном операндов и рассчитываются транслятором.

В случае, если значение данного выходит за пределы заданного шаблона, транслятор выдает предупреждающее сообщение о возможной потере некоторых разрядов результата. Усечение данных по меньшему шаблону может быть использовано с целью выделения некоторых частей цифрового значения данного.

Пример. ВЫЧИСЛИТЬ ПРИЧВИН=ВИН.

Пусть данное ПРИЧВИН содержит шесть цифр (шаблон 9(6)), из которых первые три — шифр причины, а последние три — шифр виновника простоя. Для выделения шифра виновника данное ПРИЧВИН достаточно поместить в ячейку ВИН с шаблоном 9(3).*

---

* Для выделения шифра причины и помещения его в ячейку ПРИЧ можно использовать операцию

ВЫЧИСЛИТЬ ПРИЧ=ПРИЧВИН/1000.

Шаблон данных, участвующих в вычислениях, не должен содержать литер редактирования. Литеры редактирования могут быть только в шаблоне данного, которому присваивается результат вычисленного арифметического выражения, т. е. в шаблоне данного, стоящего слева от знака равенства (при условии, что оно само не используется в качестве операнда арифметического выражения). При этом в указанном шаблоне количество литер, представляющих цифры, не должно превышать 18.

Пример.

ВЫЧИСЛИТЬ НАЛОГ=ОБЛАГАЕМАЯ * 5.5−60.5.

Данное ОБЛАГАЕМАЯ может быть описано только шаблоном цифрового класса без литер редактирования. Данное НАЛОГ может иметь, например, шаблон ПП9'руб.'99'коп.'.

Слово ОКРУГЛЯЯ в формате глагола ВЫЧИСЛИТЬ позволяет производить округление результатов вычислений в соответствии с шаблоном. При этом последняя значащая цифра, определенная шаблоном результата, увеличивается на единицу, если старшая из отбрасываемых цифр больше или равна 5. Отсутствие слова ОКРУГЛЯЯ в операторе приводит к отсечению «лишних» десятичных разрядов в соответствии с шаблоном.

Пример.

ВЫЧИСЛИТЬ А=(В+С) *Д.

Данные описаны со следующими шаблонами: шаблон А — 9(3)Т99, шаблон В — 99Т9, шаблон С — 9Т99, шаблон Д — 9Т9. Пусть до вычислений данное А имело значение 795/\04, данное В — 17/\5, данное С — 6/\44, данное Д — 8 /\ 4. В результате выполнения оператора значения данных В, С и Д не изменяются, а данное А будет иметь значение 201/\09, хотя вычисленный результат равен 201/\096.

Используя вариант ОКРУГЛЯЯ, получим А=201/\10.

Ситуация, когда количество разрядов в целой части вычисленного результата превышает количество разрядов целой части, определяемое шаблоном результата, рассматривается как переполнение.

Программист может предусмотреть в исходной программе разветвление хода выполнения рабочей программы в зависимости от наличия переполнения, используя в формате вариант ПРИ ПЕРЕПОЛНЕНИИ. При использовании этого варианта, если переполнение имело место, будет выполняться повелительный оператор, определенный форматом, а название-данного-1 сохранит прежнее значение; если переполнения не было, будет выполняться следующее предложение, а названию-данного-1 будет присвоено вычисленное значение.

Если вариант ПРИ ПЕРЕПОЛНЕНИИ не был употреблен, но произошло переполнение, старшие разряды результата будут утеряны.

Пример.

ВЫЧИСЛИТЬ ЧТС=ОКЛЧТС ПРИ ПЕРЕПОЛНЕНИИ ВЫДАТЬ 'ПОВРЕМЕНЩИК ПО ОКЛАДУ'.

Пусть шаблон данного ЧТС — 9(3), а шаблон данного ОКЛЧТС — 9(5). Пусть данное ОКЛЧТС принимает два вида значений: 1) в случае повременной оплаты по окладам — оклад, выраженный в копейках (четырех- или пятизначное число); 2) в случае повременной оплаты по часовым тарифным ставкам — часовая тарифная ставка, выраженная в копейках (трех-или двузначное число). Пусть данное ЧТС имеет шаблон 9(3). Когда данное ОКЛЧТС принимает значение первого вида, в данном ЧТС имеет место переполнение; оно сохранит свое прежнее значение, а на печать будет выдано сообщение ПОВРЕМЕНЩИК — ПО — ОКЛАДУ.

Операторы управления последовательностью выполнения процедур. В эту группу входят следующие операторы: ПЕРЕЙТИ, ИЗМЕНИТЬ, ВЫПОЛНИТЬ, ВЫЙТИ, ЕСЛИ, ОСТАНОВИТЬ.

Оператор ПЕРЕЙТИ используется для передачи управления от одной процедуры к другой.

Оператор безусловного перехода имеет формат

ПЕРЕЙТИ К название-процедуры

Иногда в некоторой точке программы в зависимости от значения переменной нужно перейти к одной из нескольких возможных процедур. Это действие можно описать при помощи оператора ПЕРЕЙТИ К название-процедуры-1, название-процедуры-2 [, название-процедуры-3 ...] В ЗАВИСИМОСТИ ОТ название-данного.

Название-процедуры может быть названием секции (без указания слова СЕКЦИЯ) или названием параграфа. В первом случае управление передается указанной процедуре программы. Во втором — той из названных в списке процедур, порядковый номер которой равен значению указанного названия данного. Последнее должно быть описано в разделе данных как цифровое данное без точки и иметь положительное значение в интервале от 1 до К, где К — количество названий процедур, перечисленных в списке ($К \leqslant 63$). Кроме того, такое данное не должно быть описано с фразой ПОВТОРЯЕТСЯ. Если значение указанного данного отрицательно, оператор не выполняется и управление передается следующему за ним оператору. Для случаев, когда значение данного окажется равным нулю или больше К, поведение программы предсказать невозможно.

Пример.

```
БЛО. ПЕРЕЙТИ К БЛ1, БЛ2, БЛ3 В ЗАВИСИМОСТИ ОТ ШНП.
БЛ1. ВЫЧИСЛИТЬ ПНАЛОГ=ОБЛАГАЕМАЯ*5.5—120,
     ВЫЧИСЛИТЬ МНАЛОГ=0, ПЕРЕЙТИ К БЛ4.
БЛ2. ВЫЧИСЛИТЬ ПНАЛОГ=ОБЛАГАЕМАЯ*3.8—480,
     ВЫЧИСЛИТЬ МНАЛОГ=ОБЛАГАЕМАЯ*.06, ПЕРЕЙТИ К БЛ4.
БЛ3. ВЫЧИСЛИТЬ ПНАЛОГ=0, ВЫЧИСЛИТЬ МНАЛОГ=0.
БЛ4. ...
```

Пусть данное ШНП (шифр налогоплательщика) принимает одно из трех значений: 1, 2, 3. В зависимости от того, каким будет значение данного ШНП к моменту выполнения процедуры БЛО, данные ПНАЛОГ (подоходный налог) и МНАЛОГ (налог на малосемейных граждан) будут рассчитываться по одной из пар формул, описанных в БЛ1, БЛ2 и БЛ3.

Оператор ИЗМЕНИТЬ используется для модификации действия предложения ПЕРЕЙТИ К без фразы В ЗАВИСИМОСТИ ОТ, выделенного в отдельный параграф.

Формат оператора:

ИЗМЕНИТЬ название-процедуры-1 ДЛЯ ПЕРЕХОДА К название-процедуры-2.

Оператор ИЗМЕНИТЬ используется для замены названия процедуры, указанного в операторе ПЕРЕЙТИ, на название-процедуры-2, указанное в операторе ИЗМЕНИТЬ. Название-процедуры-1 есть название параграфа, в котором содержится модифицируемое ПЕРЕЙТИ. Процедура-1 не должна содержать никаких других операторов, кроме единственного ПЕРЕЙТИ. Следует заметить, что если название-процедуры в операторе ПЕРЕЙТИ изменяется, то новое название-процедуры сохраняется до тех пор, пока не изменится посредством другого оператора ИЗМЕНИТЬ. Поэтому в модифицируемом операторе ПЕРЕЙТИ, вообще говоря, может быть указано любое из имеющихся в программе название процедуры при условии, что к моменту выполнения модифицируемого ПЕРЕЙТИ ему будет задано необходимое название процедуры.

Пример.

```
БЛ1. ВЫПОЛНИТЬ ПРОЦЕДУРА1, ИЗМЕНИТЬ МОДИФИКАТОР ДЛЯ
     ПЕРЕХОДА К БЛ4. ПЕРЕЙТИ К БЛ3.
БЛ2. ВЫПОЛНИТЬ ПРОЦЕДУРА2, ИЗМЕНИТЬ МОДИФИКАТОР ДЛЯ
     ПЕРЕХОДА К БЛ5.
БЛ3. ВЫПОЛНИТЬ ПРОЦЕДУРА3.
     МОДИФИКАТОР. ПЕРЕЙТИ К БЛ6.
БЛ4. ...
БЛ5. ...
БЛ6. ...
```

Если выполняется процедура БЛ1, то после выполнения БЛ3 будет выполняться оператор ПЕРЕЙТИ К БЛ4. Если выполняется процедура БЛ2, то после БЛ3 будет выполняться оператор ПЕРЕЙТИ К БЛ5.

Оператор ВЫПОЛНИТЬ обычно используют в том случае, когда некоторую группу операторов необходимо выполнить в нескольких местах КОБОЛ-программы. Вместо того чтобы выписывать в программе эти операторы каждый раз, когда они должны быть выполнены, их записывают один раз и объединяют в параграф или секцию.

Оператор ВЫПОЛНИТЬ обеспечивает временное отклонение от последовательного выполнения процедур для того, чтобы выполнить параграф или секцию указанное количество раз, или до тех пор, пока не будет выполнено некоторое условие.

Формат оператора:

Вариант 1

ВЫПОЛНИТЬ название-процедуры-1 [ПО название-процедуры-2]

$$\left[\left\{\begin{array}{l}\text{целое}\\ \text{название-данного}\end{array}\right\}\quad\left\{\begin{array}{l}\underline{\text{РАЗ}}\\ \underline{\text{РАЗА}}\end{array}\right\}\right]$$

Вариант 2

ВЫПОЛНИТЬ название-процедуры-1 [ПО название-процедуры-2] ДО условие

Вариант 3

ВЫПОЛНИТЬ название-процедуры-1 [ПО название-процедуры-2

МЕНЯЯ { название-индекса-1 / название-данного-1 } ОТ { название-индекса-2 / числовой-литерал-2 / название-данного-2 }

НА { числовой-литерал-3 / название-данного-3 } ДО условие-1

[МЕНЯЯ { название-индекса-4 / название-данного-4 } ОТ { название-индекса-5 / числовой-литерал-5 / название-данного-5 }

НА { числовой-литерал-6 / название-данного-6 } ДО условие-2]

Каждое название-процедуры — это название секции или параграфа в разделе процедур.

Каждое название-данного представляет собой цифровое элементарное данное, описанное в разделе данных как вычислительное данное с шаблоном, содержащим не более девяти цифр. Каждое название-индекса определено своим появлением в разделе данных во фразе ИНДЕКСИРУЕТСЯ название-индекса.

Значение названия-данного в варианте 1 должно быть неотрицательным числом. Если оно равно 0, то выполнения процедуры не произойдет.

Если в варианте 3 меняются два параметра, они должны быть одинакового типа: оба названия индексов или названия данных.

Если в варианте 3 меняется название-данного-1 (название-данного-4), то начальное значение может быть задано только числовым-литералом-2 или названием-данного-2 (числовым-литералом-5 или названием-данного-5) и числовой-литерал-3 или название-данного-3 (числовой-литерал-6 или название-данного-6) должны иметь положение десятичной точки, соответствующее названию-данного-1 (названию-данного-4), а количество цифр не больше, чем в названии-данного-1 (названии-данного-4).

Глагол ВЫПОЛНИТЬ передает управление первому оператору процедуры, имеющей название название-процедуры-1, и обеспечивает возврат управления к оператору, следующему за ВЫПОЛНИТЬ.

Если название-процедуры-1 есть название параграфа и название-процедуры-2 не указано, то возврат происходит после выполнения последнего оператора этого параграфа.

Если название-процедуры-1 есть название секции, а название-процедуры-2 не указано, то возврат происходит после выпол-

нения последнего оператора последнего параграфа этой секции.

Если название-процедуры-2 определено, то возврат управления к следующему за ВЫПОЛНИТЬ высказыванию происходит после выполнения последнего оператора процедуры (секции или параграфа), названной названием-процедуры-2.

Не следует оканчивать выполняемый участок, т. е. область действия глагола ВЫПОЛНИТЬ, глаголом ПЕРЕЙТИ. В противном случае управление будет передано не оператору, следующему за ВЫПОЛНИТЬ, а процедуре, указанной в операторе ПЕРЕЙТИ.

Все ограничения на использование операторов ПЕРЕЙТИ и ВЫПОЛНИТЬ в области действия глагола ВЫПОЛНИТЬ связаны с тем, что в рабочей программе в конце области действия автоматически формируется команда возврата либо на оператор, следующий за ВЫПОЛНИТЬ (в случае однократного действия глагола ВЫПОЛНИТЬ), либо на повторное действие ВЫПОЛНИТЬ (в случае циклического действия глагола ВЫПОЛНИТЬ). Поэтому для обеспечения такого возврата управление внутри выполняемого участка должно быть передано последнему оператору процедуры-2.

Пример.

```
ПР1. ВЫПОЛНИТЬ ПР2 ПО ПР5.
ПР2. ПЕРЕЙТИ К ПР3, ПР4 В ЗАВИСИМОСТИ ОТ ПЕРЕКЛЮЧАТЕЛЬ.
ПР3. ВЫЧИСЛИТЬ КАТАЛОГ=КАТАЛОГ1+КАРТА. ПЕРЕЙТИ К ПР5.
ПР4. ВЫЧИСЛИТЬ КАТАЛОГ=КАТАЛОГ2+КАРТА *2.
ПР5. ВЫЧИСЛИТЬ ОБЩФОНД=ФОНД+КАТАЛОГ.
```

Оператор ПЕРЕЙТИ может быть использован для преждевременного выхода из области действия ВЫПОЛНИТЬ.

Пример.

```
ПР1. ВЫПОЛНИТЬ ПР2 ПО ПР5. ПЕРЕЙТИ К ПР3.
                .
                .
                .
ПР2. ЧИТАТЬ СПРАВОЧНЫЙ В КОНЦЕ ПЕРЕЙТИ К ПР4.
ПР5. ...
```

В данном примере еще до окончания действия глагола ВЫПОЛНИТЬ может быть достигнут конец массива СПРАВОЧНЫЙ, в результате чего произойдет переход из области действия ВЫПОЛНИТЬ к процедуре ПР4, где обрабатывается эта «критическая» ситуация.

Если область действия оператора ВЫПОЛНИТЬ включает в себя другой оператор ВЫПОЛНИТЬ, то их области действия не должны частично пересекаться. Другими словами, область действия внутреннего ВЫПОЛНИТЬ должна быть либо полностью исключена из области действия внешнего, либо полностью содержаться в ней. Во втором случае последние параграфы областей действия двух ВЫПОЛНИТЬ не должны совпадать (см. выше замечание о ячейке с командой возврата).

Пример.

```
                ВЫПОЛНИТЬ ПАРАГРАФ1 ПО ПАРАГРАФ4. ПЕРЕЙТИ
                  К ПАРАГРАФ6.
ПАРАГРАФ1. ВЫПОЛНИТЬ ПАРАГРАФ2 ПО ПАРАГРАФ3. ПЕРЕЙТИ
                К ПАРАГРАФ5.
ПАРАГРАФ2. ВЫЧИСЛИТЬ СУММА1=СУММА2+ПОПРАВКА.
ПАРАГРАФ3. ВЫЧИСЛИТЬ СУММА3=СУММА1*ПРОЦЕНТ.
ПАРАГРАФ9. ВЫЧИСЛИТЬ СУММА4=СУММА2*ПРОЦЕНТ.
ПАРАГРАФ5. ВЫЧИСЛИТЬ СУММА5=(СУММА2+СУММА1)*ПРОЦЕНТ1.
                ВЫЧИСЛИТЬ СУММА6=(СУММА2+СУММА1)*ПРОЦЕНТ2.
ПАРАГРАФ4. ПОМЕСТИТЬ СУММА5 В СУМПЕЧ.
                ПОМЕСТИТЬ СУММА6 В СУМПЕЧ1.
ПАРАГРАФ8. ВЫЧИСЛИТЬ СУММА7=(СУММА2+СУММА1)* ПРОЦЕНТ3.
                ПОМЕТИТЬ СУММА7 В СУМПЕЧ2.
                ПЕРЕЙТИ К ПАРАГРАФ7.
ПАРАГРАФ6. ВЫЧИСЛИТЬ СУММА8=(СУММА2+СУММА1)*ПРОЦЕНТ4.
                ПОМЕСТИТЬ СУММА8 В СУМПЕЧ2.
ПАРАГРАФ7. ВЫЧИСЛИТЬ СУММА5=СУММА3+СУММА1.
                ВЫЧИСЛИТЬ СУММА6=(СУММА3+СУММА1)* ПРОЦЕНТ1.
                ВЫЧИСЛИТЬ СУММА7=(СУММА3+СУММА1) *ПРОЦЕНТ2.
```

Приведенный ниже пример иллюстрирует один из типичных случаев неправильного использования одного глагола ВЫПОЛНИТЬ в области действия другого.

Пример.

```
                ВЫПОЛНИТЬ ЭТАП1 ПО ФИНАЛ. ...
ЭТАП1.          ВЫЧИСЛИТЬ ОЧКИ1=ОЧКИ1+СУММА*5.5.
                ВЫПОЛНИТЬ ЭТАП2 ПО ПОЛУФИНАЛ2. ПЕРЕЙТИ К
                ФИНАЛ.
ЭТАП2.          ВЫЧИСЛИТЬ ОЧКИ2=ОЧКИ2+2*СУММА1+СУММА2.
ФИНАЛ.          ВЫЧИСЛИТЬ ОБЩОЧКИ=ОБЩПОЛУ1+ОБЩПОЛУ2. ...
ПОЛУФИНАЛ2. ВЫЧИСЛИТЬ ОБЩПОЛУ2=ОБЩПОЛУ2+ОЧКИ2.
```

Здесь произойдет возврат на оператор, следующий за первым ВЫПОЛНИТЬ, в процессе действия второго ВЫПОЛНИТЬ (как только второе ВЫПОЛНИТЬ достигнет параграфа ФИНАЛ). Таким образом, второе ВЫПОЛНИТЬ никогда не будет **выполнено до конца.**

Области действия нескольких ВЫПОЛНИТЬ могут пересекаться только в случае, если никакое из ВЫПОЛНИТЬ не содержится в области действия другого ВЫПОЛНИТЬ.

Процедуры, входящие в область действия некоторого ВЫПОЛНИТЬ, могут быть использованы и без ВЫПОЛНИТЬ. Если управление передается этим процедурам не с помощью ВЫПОЛНИТЬ, то вслед за последней процедурой из области действия ВЫПОЛНИТЬ будет выполняться следующая за ней процедура, как если бы эти процедуры вообще не упоминались в операторе ВЫПОЛНИТЬ. Если такая область действия была выполнена (по глаголу ВЫПОЛНИТЬ) до конца, то в ячейке возврата, следующей за областью действия глагола ВЫПОЛНИТЬ в рабочей программе, содержится нуль. В процессе выполнения рабочей программы «нулевая» команда пропускается. Если при действии глагола ВЫПОЛНИТЬ был сделан преждевременный выход из области действия, то ячейка с командой

возврата не будет очищена и передавать управление этой области без ВЫПОЛНИТЬ нельзя.

Следует иметь в виду, что если процедура-1 непосредственно следует за оператором ВЫПОЛНИТЬ название-процедуры-1 целое РАЗ, то фактически она будет выполнена на один раз больше, чем указано в формате.

Пример.

```
                ВЫПОЛНИТЬ ПРОТЯЖКА 5 РАЗ.
ПРОТЯЖКА.       ВЫДАТЬ ПРОБЕЛ
```

Здесь параграф ПРОТЯЖКА будет выполнен 6 раз (5 раз с помощью ВЫПОЛНИТЬ и 1 раз в естественном порядке после этого оператора).

Вариант 2 указывает, что заданные процедуры должны выполняться до тех пор, пока не будет выполнено (не станет истинным) условие, определенное после слова ДО, при этом управление будет передано следующему за ВЫПОЛНИТЬ оператору. Если условие окажется истинным в самом начале, то выполнения процедуры не произойдет.

Пример.

```
    ПОМЕСТИТЬ 0 В ЧИСЛО. ПОМЕСТИТЬ 1 В ФАКТОРИАЛ.
    ВЫПОЛНИТЬ БЛОК1 ПО БЛОК3 ДО ФУНКЦИЯ<.001.
    ПЕРЕЙТИ К БЛОК4.
БЛОК1. ВЫЧИСЛИТЬ ЧИСЛО=ЧИСЛО+1.
БЛОК2. ВЫЧИСЛИТЬ ФАКТОРИАЛ=ЧИСЛО  ФАКТОРИАЛ.
БЛОК3. ВЫЧИСЛИТЬ ФУНКЦИЯ=1/ФАКТОРИАЛ.
БЛОК4. ...
```

Здесь процедуры БЛОК1÷БЛОК3 будут выполняться циклически до тех пор, пока значение данного ФУНКЦИЯ не станет меньше числа 0,001, после чего управление будет передано процедуре БЛОК4.

Вариант 3 используется, если при каждом выполнении указанных в формате процедур значения одного или двух названий-данных или названий-индексов должны получать некоторое приращение. В следующем ниже описании варианта МЕНЯЯ все, что касается названий-данных, относится также и к названиям-индексов. Слово ОТ оказывает на название-индекса такое же действие, что и оператор УСТАНОВИТЬ.

При выполнении оператора с вариантом МЕНЯЯ, когда изменяется одно название-данного, вначале названию-данного-1 присваивается указанное начальное значение (название-данного-2 или числовой-литерал-2). Затем проверяется условие. Если условие неверно, происходит выполнение процедуры, после чего изменяется название-данного-1 на указанное приращение (название-данного-3 или числовой-литерал-3) и вновь проверяется условие. Это повторяется до тех пор, пока условие не станет истинным, после чего управление получает следующий за ВЫПОЛНИТЬ оператор.

Пример.

```
    ПОМЕСТИТЬ 1 В ФАКТОРИАЛ.
    ВЫПОЛНИТЬ БЛОК1 МЕНЯЯ ЧИСЛО ОТ 1 НА 1
    ДО ФУНКЦИЯ<.001. ПЕРЕЙТИ К БЛОК2.
```

```
БЛОК1. ВЫЧИСЛИТЬ ФАКТОРИАЛ=ЧИСЛО*ФАКТОРИАЛ.
       ВЫЧИСЛИТЬ ФУНКЦИЯ=1/ФАКТОРИАЛ.
БЛОК2. ...
```

Процедура БЛОК1 будет выполняться подобно тому, как это происходит в предыдущем примере. При этом данное ЧИСЛО будет принимать последовательный ряд значений за счет использования слова МЕНЯЯ.

Данный пример представляет нетипичный случай использования варианта 3. Как правило, вариант со словом МЕНЯЯ используется для модификации значений индексов.

Пример.

```
       ПОМЕСТИТЬ 0 В СУММА, ОБЩАЯ.
       ВЫПОЛНИТЬ ПРОЦ1 МЕНЯЯ ИНД ОТ 1 НА 1 ДО ИНД=13.
ПРОЦ1. ЕСЛИ МЕС(ИНД)=МЕСЯЦ ВЫЧИСЛИТЬ СУММА=СУММА+
       +СУМ(ИНД).
       ВЫЧИСЛИТЬ ОБЩАЯ=ОБЩАЯ+СУМ(ИНД).
```

Здесь выполняется поиск месяца, заданного данным МЕСЯЦ в связанном поле следующей структуры:

```
01 СПИСОК.
02 СТРОКА ПОВТОРЯЕТСЯ 12 РАЗ ИНДЕКСИРУЕТСЯ ИНД.
03 МЕС Ш 99.
03 СУМ Ш 9(5)Т99.
```

После того как группа с заданным месяцем обнаружена, производится вычисление с данным СУМ из найденной группы — вычисляется данное СУММА. Одновременно в процессе поиска в списке заданного месяца производится накопление данного СУМ по всем значениям индекса — вычисляется данное ОБЩАЯ.

Если в варианте МЕНЯЯ изменяются два названия-данного (название-данного-1 или название-данного-4), то сначала названию-данного-1 и названию-данного-4 присваиваются их начальные значения (значения названия-данного-2 и названия-данного-5 соответственно). Затем проверяется условие-1. Если оно истинно, то управление получает следующий за ВЫПОЛНИТЬ оператор; если ложно, то проверяется условие-2. Если условие-2 ложно, происходит выполнение процедуры, указанной в формате, после чего название-данного-4 получает приращение, соответствующее названию-данного-6, и условие-2 проверяется снова. Это повторяется до тех пор, пока условие-2 не станет истинным. Когда условие-2 истинно, названию-данного-4 опять присваивается начальное значение (значение названия-данного-5), название-данного-1 получает приращение, соответствующее названию-данного-3, и проверяется условие-1. Работа ВЫПОЛНИТЬ окончена, если условие-1 истинно; если нет, этот цикл продолжается до тех пор, пока условие-1 не станет истинным.

Пример.

Пусть требуется умножить матрицу Е на вектор Д, получая результирующий вектор Ф:

$$Д = \begin{pmatrix} Д1 \\ Д2 \\ Д3 \end{pmatrix}; \quad Е = \begin{pmatrix} Е11 & Е12 & Е13 \\ Е21 & Е22 & Е23 \\ Е31 & Е32 & Е33 \end{pmatrix}; \quad Ф = \begin{pmatrix} Ф1 \\ Ф2 \\ Ф3 \end{pmatrix},$$

где Ф1=Е11·Д1+Е12·Д2+Е13·Д3;
Ф2=Е21·Д1+Е22·Д2+Е23·Д3;
Ф3=Е31·Д1+Е32·Д2+Е33·Д3.

Опишем заданные векторы и матрицу следующим образом:

```
01 ВЕКТОР.
02 Д Ш 9 ПОВТОРЯЕТСЯ 3 РАЗА, ИНДЕКСИРУЕТСЯ ИНД3.
01 МАТРИЦА.
02 СТРОКА ПОВТОРЯЕТСЯ 3 РАЗА ИНДЕКСИРУЕТСЯ ИНД1.
03 Е Ш 9 ПОВТОРЯЕТСЯ 3 РАЗА ИНДЕКСИРУЕТСЯ ИНД2.
01 РЕЗУЛЬТАТ.
02 Ф Ш 9 ПОВТОРЯЕТСЯ 3 РАЗА, ИНДЕКСИРУЕТСЯ ИНД4.
```

Результирующий вектор можно получить с помощью следующей группы операторов.

```
    ПОМЕСТИТЬ О В Ф(1), Ф(2), Ф(3).
    ВЫПОЛНИТЬ БЛ1 МЕНЯЯ ИНД1 ОТ 1 НА 1 ДО ИНД1=4.
    ПЕРЕЙТИ К БЛ3.
БЛ1. УСТАНОВИТЬ ИНД4 НА ИНД1.
    ВЫПОЛНИТЬ БЛ2 МЕНЯЯ ИНД2 ОТ 1 НА 1 ДО ИНД2=4.
БЛ2. УСТАНОВИТЬ ИНД3 НА ИНД2.
    ВЫЧИСЛИТЬ Ф(ИНД4)=Ф(ИНД4)+Е(ИНД1, ИНД2)*Д(ИНД3).
БЛ3. ...
```

Оператор ВЫЙТИ обеспечивает общую точку выхода для ряда процедур. Необходимость в такой точке возникает в случае, если в области действия ВЫПОЛНИТЬ имеются разветвляющиеся процедуры, приводящие выполняемый процесс к нескольким конечным точкам. Чтобы охватить такой ветвящийся процес действием одного глагола ВЫПОЛНИТЬ, необходимо свести все конечные процедуры этого процесса к одной точке. Такой точкой должен быть оператор ВЫЙТИ, образующий отдельное предложение, перед которым должно стоять название параграфа.

Формат оператора:

```
ВЫЙТИ.
```

Оператор ВЫЙТИ должен быть единственным оператором параграфа.

В рабочей программе оператор ВЫЙТИ реализуется пустой ячейкой, где формируется команда возврата из области действия ВЫПОЛНИТЬ.

Если управление передано ВЫЙТИ без участия ВЫПОЛНИТЬ, то ВЫЙТИ пропускается, т. е. управление передается процедуре, написанной за ВЫЙТИ.

Например, пусть имеется документ следующей структуры:

```
01 СПИСОК.
02 СТРОКА-СПИСКА ПОВТОРЯЕТСЯ 50 РАЗ ИНДЕКСИРУЕТСЯ ИНД.
03 Н-ЗАЧ-КН Ш 9(7).
03 ПРЕДМ Ш Х(12).
03 ОЦЕНК Ш 9.
```

Не все строки списка заполнены: начиная с некоторой строки и до конца идут строки, в которых значения данных равны нулю. Необходимо найти первую пустую строку в списке и за-

нести в нее данные НОМЕР-КНИЖКИ, ПРЕДМЕТ и ОЦЕНКА при условии, что такая комбинация данных в списке раньше не встречалась. В противном случае, а также если в списке нет пустых строк, данные НОМЕР-КНИЖКИ, ПРЕДМЕТ и ОЦЕНКА в список заносить не следует.

Поставленную задачу можно выполнить с помощью следующей группы операторов:

```
        ВЫПОЛНИТЬ БЛ1 ПО ВЫХ МЕНЯЯ ИНД ОТ 1
        НА 1
        ДО ИНД-51. ПЕРЕЙТИ К БЛ3.
БЛ1.    ЕСЛИ Н-ЗАЧ-КН(ИНД) НЕ=НОМЕР-КНИЖКИ
        ПЕРЕЙТИ К БЛ2.
        ЕСЛИ ПРЕДМ(ИНД) НЕ=ПРЕДМЕТ ПЕРЕЙТИ
        К БЛ2.
        ЕСЛИ ОЦЕНК(ИНД) НЕ=ОЦЕНКА ПЕРЕЙТИ
        К БЛ2.
        ПЕРЕЙТИ К ВЫХ.
БЛ2.    ЕСЛИ Н-ЗАЧ-КН (ИНД)=О ПОМЕСТИТЬ НОМЕР-
        КНИЖКИ В Н-ЗАЧ-КН(ИНД), ПОМЕСТИТЬ
        ПРЕДМЕТ В ПРЕДМ(ИНД),
        ПОМЕСТИТЬ ОЦЕНКА В ОЦЕНК(ИНД).
ВЫХ.    ВЫЙТИ.
БЛ3.    ...
```

Оператор ЕСЛИ обеспечивает выбор действий в зависимости от выполнения (истинности) или невыполнения (ложности) некоторого условия.

Формат оператора:

ЕСЛИ условие { ; повелительный-оператор-1 / СЛЕДУЮЩЕЕ ПРЕДЛОЖЕНИЕ }

[; ИНАЧЕ { ; повелительный-оператор-2 / ; СЛЕДУЮЩЕЕ ПРЕДЛОЖЕНИЕ }]

Если условие истинно, выполняется повелительный-оператор-1 (следующее предложение), если ложно — повелительный-оператор-2 (следующее предложение).

Условие в формате может быть простым и сложным.

Существует три вида простых условий:

проверка отношения (сравнение);
проверка класса данного;
проверка типа записи.

Условие вида «проверка отношения» имеет формат:

{ название-индекса-1 / название-данного-1 / литерал-1 / арифметическое-выражение-1 } [НЕ] { БОЛЬШЕ (>) / МЕНЬШЕ (<) / РАВНО (=) } { название-индекса-2 / название-данного-2 / литерал-2 / арифметическое-выражение-2 }

Литерал не может быть употреблен в обеих частях условия одновременно.

Названия-данных в формате могут быть названиями элементарных или групповых данных. Сравнивать можно элементарное данное с элементарным, групповое с групповым. Сравниваемые групповые данные должны иметь одинаковый размер и структуру. Номера уровней, а также уровневая структура групповых данных не играют роли при сравнении. Значение имеет только число элементарных данных в сравниваемых группах: оно должно быть одинаковым.

Литерал и арифметическое выражение рассматриваются при сравнении как элементарные данные.

При сравнении элементарных данных сравниваемые величины должны быть совместимы по классу, т. е. нельзя, например, сравнивать цифровую величину с буквенной. Допускаются сравнения цифровых величин с цифровыми, буквенно-цифровых с буквенно-цифровыми и буквенными, цифровых редактируемых величин с цифровыми редактируемыми.

Нечисловой литерал рассматривается как данное буквенно-цифрового класса. Класс (внутреннее представление) числового литерала приводится в соответствие с классом второго операнда сравнения.

Для цифровых величин проверка отношения состоит в установлении, какая из них меньше, равна или больше другой. Длина операндов (количество литер) не играет роли при сравнении. Так, величины 45, 00045 и + 45∧000 считаются равными. При сравнении учитывается знак цифровой величины: всякое положительное число больше любого отрицательного числа, отрицательное число с меньшей абсолютной величиной больше отрицательного числа с большей абсолютной величиной и т. д.

Для нецифровых величин сравнение идет посимвольно в соответствии с расположением символов в таблице кодов машины «Минск-32».

Нуль считается наименьшей величиной, пробел — больше любой из цифр, но меньше любой из букв, любая буква больше любой цифры и т. д.

Сравнение начинается с самой левой литеры данного. Если оказывается, что эти литеры совпадают, сравниваются следующие литеры и т. д. Если сравнение каждой пары литер дает равенство и сравниваются величины одинаковой длины, то они равны. Как только нарушается совпадение литер, сравнение прекращается, а для несравнившихся литер определяется их положение в таблице кодов машины. Та из литер больше, которая дальше отстоит от начала таблицы, и та из величин будет больше, у которой раньше встретилась «большая» литера.

Например, при сравнении СТОЛЫ и СТУЛ получится, что вторая величина больше, так как У «больше» О. При сравнении величин разной длины меньшая дополняется пробелами до раз-

мера большей величины и сравнение происходит так же, как и сравнение величин равной длины.

Так, из двух буквенных величин «наряд» и «наряды» вторая больше, так как «ы» «больше» пробела, «наряд» и «наряд— были бы признаны равными, так как дополняющая литера в более короткой величине является пробелом.

Сравнение групповых данных выполняется поячеечно. Если размеры данных не совпадают, при трансляции выдается сообщение об ошибке.

Участвующие в сравнении групповые данные из секции рабочей памяти не должны переопределять поля большого размера, так как позиции, не занятые значением переопределяющего данного, могут сохранять значения переопределяемого данного. При поячеечном сравнении это может привести к несравнению фактически равных групповых данных. Однако это возможно, если предварительно заполнить нулями поле, в которое будет помещен групповой пункт, участвующий в сравнении, или применить поэлементарное сравнение групповых пунктов. Сравниваемые групповые данные не должны содержать символов, двоичные коды которых более кода 0111111. Если такие символы встретятся в позициях 5к (к=0, 1,2), т. е. будут последними 7-битовыми символами ячейки, то результат сравнения будет неправильный.

Поскольку данные, используемые для индексирования, преобразуются в специфическую, индексную форму представления, не все из них могут сравниваться с любыми цифровыми величинами. Так, индекс, описанный в разделе данных, сравнивается без преобразования в вычислительную форму и потому может сравниваться с индексом, определенным во фразе ИНДЕКСИРУЕТСЯ, или с другим индексом, описанным в разделе данных. Индекс же, определенный во фразе ИНДЕКСИРУЕТСЯ, преобразуется в соответствии с формой представления второго члена отношения и может сравниваться с любыми цифровыми величинами.

Таблица 4.1

Тип левого операнда сравнения	Тип правого операнда сравнения			
	н-д (для вычисления)	н-д-1 (для индекса)	н-и	ч-л
н-д (для вычислений)	да	—	да	да
н-д-1 (для индекса)	—	да	да	—
н-и	да	да	да	да
ч-л	да	—	да	—

Допустимые сочетания аргументов сравнения, включающего названия индексов и (или) индексных пунктов данных, приведены в табл. 4. 1.

Необходимо подчеркнуть, что на представление данных оказывает влияние не фраза описания ДЛЯ ВЫЧИСЛЕНИЙ или ДЛЯ ИНДЕКСА, а сам факт их использования для вычисления или для индексирования. Фразы же об использовании имеют лишь комментарное значение.

Пример.

ВЫЧИСЛИТЬ МИЛЛИОН=МИЛЛИАРД.
ВЫЧИСЛИТЬ МИЛЛИОНЫ=МИЛЛИАРД—МИЛЛИОН.
ЕСЛИ МИЛЛИОНЫ НЕ=0 ВЫДАТЬ МИЛЛИОНЫ, 'МИЛЛИОНОВ'.

Здесь данное МИЛЛИОН имеет шаблон 9(6), данное МИЛЛИАРД — шаблон 9(9), данное МИЛЛИОНЫ — шаблон 9(9). Если в результате проверки условия окажется, что три старших разряда данного МИЛЛИАРД не равны нулю, на печать будет выдано число МИЛЛИОНЫ и слово МИЛЛИОНОВ.

Условие вида «проверка класса» имеет формат

название-данного [НЕ] { ЦИФРОВОЕ / БУКВЕННОЕ }

Данное относится к цифровому классу, если оно состоит только из цифр, знака и десятичной точки, и к буквенному, если оно состоит из букв и пробелов. Данное, класс которого проверяется, не должно быть описано как цифровое. Его шаблон должен состоять только из символов X.

Пример.

Пусть имеется документ следующего вида:

01 СЛОВАРЬ.
02 ОСНОВА Ш_А(7).
02 ПР-МЯГКОСТИ Ш_X.

Здесь данное ОСНОВА может принимать одно из значений: январ, феврал, март, апрел, июн, июл, август, сентябр, октябр, ноябр, декабр. Данное ПР-МЯГКОСТИ содержит значение признака мягкости основы: ь — если соответствующее название месяца оканчивается на мягкий знак, и 0 (нуль) — в противном случае. Необходимо образовать родительный падеж от заданной основы и выдать его на печать.

Задача решается путем использования следующего оператора:

ЕСЛИ ПР-МЯГКОСТИ ЦИФРОВОЕ ВЫДАТЬ ОСНОВА, 'А',
ИНАЧЕ ВЫДАТЬ ОСНОВА, 'Я'.

Условие вида «проверка типа записи» имеет формат

ЗАПИСЬ название-записи

Такая проверка имеет смысл в случае, если обрабатываемый массив состоит из записей нескольких типов и необходимо выполнить те или иные действия в зависимости от типа прочитанной записи.

Условие считается выполненным, если шифр записи, находящейся в момент проверки условия в зоне записи, совпадает с шифром, указанным в описании записи, названной в условии.

ЕСЛИ ЗАПИСЬ ТАБЕЛЬ ПЕРЕЙТИ К РАСЧЕТ-ТАБЕЛЯ.
ЕСЛИ ЗАПИСЬ НАРЯД ПЕРЕЙТИ К РАСЧЕТ-НАРЯДА.
ЕСЛИ ЗАПИСЬ ДОПЛАТА ПЕРЕЙТИ К РАСЧЕТ-ДОПЛАТЫ.

Сложное условие образуется путем соединения простых условий логическими операторами И и ИЛИ, означающими соответственно операции «логического умножения» и «логического сложения».

Таблица 4.2

Значение условия		Логическая операция и результат ее выполнения		
условие-1	условие-2	условие-1 И условие-2	условие-1 ИЛИ условие-2	НЕ условие-1
истина	истина	истина	истина	ложь
истина	ложь	ложь	истина	ложь
ложь	ложь	ложь	ложь	истина
ложь	истина	ложь	истина	истина

В следующей ниже табл. 4.2 показано отношение между исходными значениями условий, связанных логическими операторами, и результатами выполнения логических операций.

Для определения порядка выполнения логических операций могут быть использованы круглые скобки. Если порядок операций не определяется скобками, сначала выполняется операция И, затем — операция ИЛИ.

Например, условие

ТАБНОМЕР=0 и ЦЕХ=0 ИЛИ КАТЕГОРИЯ=0 и ВИДРАБОТ=0 ИЛИ СИСТЕМАОПЛ=0 эквивалентно следующему:

(ТАБНОМЕР=0 и ЦЕХ=0) ИЛИ (КАТЕГОРИЯ=0 И ВИДРАБОТ=0) ИЛИ СИСТЕМАОПЛ=0.

Пример.

```
    ВЫПОЛНИТЬ БЛ1 ПО ВЫХ МЕНЯЯ ИНД ОТ 1 НА 1
    ДО ИНД=51. ПЕРЕЙТИ К БЛ3.
БЛ1. ЕСЛИ Н-ЗАЧ-КН(ИНД) НЕ =НОМЕР-КНИЖКИ И
            ПРЕДМ(ИНД) НЕ=ПРЕДМЕТ И
            ОЦЕНК(ИНД) НЕ=ОЦЕНКА ПЕРЕЙТИ К БЛ2.
    ПЕРЕЙТИ К ВЫХ.
БЛ2. ЕСЛИ Н-ЗАЧ-КН(ИНД)=0 ПОМЕСТИТЬ НОМЕР-КНИЖКИ
      В Н-ЗАЧ-КН(ИНД), ПОМЕСТИТЬ ПРЕДМЕТ В ПРЕДМ(ИНД),
      ПОМЕСТИТЬ ОЦЕНКА В ОЦЕНК(ИНД).
ВЫХ. ВЫЙТИ.
БЛ3. ...
```

Оператор ОСТАНОВИТЬ прекращает выполнение рабочей программы временно или окончательно.

Формат оператора: ОСТАНОВИТЬ { нечисловой-литерал / РАБОТУ }

Вариант «ОСТАНОВИТЬ нечисловой-литерал» используется для указания промежуточного останова с ожиданием ответа оператора. При этом происходит вывод указанного в формате литерала на пишущую машинку. Продолжение выполнения рабочей программы начинается со следующего за ОСТАНОВИТЬ предложения после ответа оператора.

Для прекращения выполнения программы используется вариант ОСТАНОВИТЬ РАБОТУ. При выполнении этого оператора происходит передача управления ДИСПЕТЧЕРу.

Вариант ОСТАНОВИТЬ РАБОТУ следует использовать только в головной программе дерева программ, составляющих задачу. Для обеспечения возврата во внешнюю программу необходимо использовать оператор ВЫЙТИ ИЗ-ПРОГРАММЫ.

Оператор ОСТАНОВИТЬ должен быть последним или единственным в повелительном предложении или в группе повелительных операторов условного предложения.

Операторы перемещения данных. К этой группе относятся операторы: ПОМЕСТИТЬ, ПРОСМОТРЕТЬ, УСТАНОВИТЬ.

Оператор ПОМЕСТИТЬ служит для перемещения данных из одного места памяти в другое. При этом данные в исходном поле сохраняют прежнее значение.

Формат оператора:

ПОМЕСТИТЬ { название-данного-1 / литерал / фигуральная константа / СЧЕТЧИК } В { название-данного-2 / СЧЕТЧИК }

[, { название-данного-3 / СЧЕТЧИК } ...]

Название-данного-1 называется посылаемым полем, а название-данного-2 (название-данного-3) принимающим.

Таблица 4.3

Класс посылаемого данного	Класс принимающего данного			
	цифровой	буквенный	буквенно-цифровой	редактируемый цифровой
Цифровой	да	—	да	да
Буквенный	—	да	да	—
Буквенно-цифровой	да	да	да	—
Редактируемый цифровой	—	—	да	да*)

* Допускается только пересылка в данное с таким же шаблоном, что и шаблон посылаемого данного.

Можно пересылать элементарное данное в элементарное, групповое в групповое, причем в случае пересылки элементарного данного в элементарное классы данных должны быть совместимы (табл. 4.3), а в случае пересылки группового данного в групповое описания данных должны совпадать. В противном случае при трансляции будет выдаваться сообщение об ошибке.

В формате могут быть использованы фигуральные константы НУЛЬ, ПРОБЕЛ, КАВЫЧКА, ВСЕ нечисловой-литерал для пересылки их в групповые и элементарные данные. Поле, отведенное под групповое данное, заполняется при этом сплошь соответствующими фигуральной константе символами.

Элементарные данные при пересылке приводятся в соответствие с шаблоном принимающего поля.

При пересылке цифровых величин в цифровые происходит их выравнивание по десятичной точке. При этом целые и дробные части величин помещаются независимо друг от друга: целые помещаются, начиная с первого символа справа, дробные — слева. Если количество цифр в целой части посылаемого поля больше того, которое указано шаблоном целой части принимающего поля, транслятор выдает сообщение о возможной потере старших разрядов пересылаемого данного. При потере десятичных разрядов после десятичной точки предупреждение не выдается.

Пример.
Пусть помещаемое данное имеет шаблон 9(4)Т99 и значение 123/\05, и принимающее имеет шаблон 9Т9. В результате помещения в принимающем поле получим число 3/\0.

Если количество цифр в посылаемом данном меньше того, которое указано шаблоном принимающего данного, посылаемое данное дополняется нулями слева и / или справа в соответствии с шаблоном принимающего поля.

Пример.
Пусть помещаемое данное имеет шаблон 99 и значение 85; пусть шаблон принимающего данного 9(3)Т9. Результат помещения — 085/\0.

Если это требуется, данное редактируется. При этом следует помнить, что вставляемые нечисловые литералы, являющиеся наименованиями единиц измерения типа «км», «руб», «коп», в шаблоне не эквивалетны десятичной точке. Если, например, посылаемое поле описано как 999Т99 и имеет значение 132/\81, то после пересылки в поле с шаблоном 999 'руб.' 99 'коп' оно будет отредактировано к виду 001 руб. 32 коп, а не 132 руб. 81 коп. Можно получить результат в виде 132 руб. 81 коп, например, задав шаблон принимающего поля как 999 'руб'. 99 'коп', т. е. с действительной десятичной точкой. При этом происходит выравнивание по действительной десятичной точке в принимающем поле и неявной десятичной точке в помещаемом данном.

При пересылке данных буквенного и буквенно-цифрового классов в данные этих же классов избыток позиций в принимающем поле заполняется пробелами справа.

Если принимающее поле не может вместить посылаемое данное, транслятор выдает предупреждающее сообщение. При выполнении рабочей программы пересылается, начиная слева, столько символов, сколько помещается в принимающем поле.

При помещении данных цифрового класса в буквенно-цифровой дополнение и усечение пересылаемых данных происходит по шаблону принимающего данного.

Примеры пересылки данных разных классов:

Посылаемое поле		Принимающее поле	
шаблон	значение данного	шаблон	результат пересылки
9 (3)	123	X (3)	123
9 (3)	285	X (2)	28
9 (2)	43	X (5)	43— — —
X (4)	5725	9 (3)	725
X (3)	149	9 (4)	0149

Позиции, дополняющие поле, занимаемое данным, до целого числа ячеек, заполнены нулями.

Примеры пересылки элементарного данного в элементарное:

Посылаемое поле		Принимающее поле	
шаблон	значение данного	шаблон	результат пересылки
99T999	72∧435	99T999	72∧435
99T9 (3)	25∧213	9 (3) T9 (2)	025∧21
999T9	523∧5	9 (3) T9 (3)	523∧500
9 (3) T9 (2)	407∧98	9 (2) T9 (2)	07∧98*
99T9	85∧6	9 (3)	085
9 (3) T9	001∧3	ПП9.9 (2)	— — 1.30
A (6)	ДОКЛАД	A (4)	ДОКЛ
A (4)	ЛИСТ	X (8)	ЛИСТ — — — —
99999	92316	999'м'99'см'	923 м 16 см

* Транслятором будет выдано сообщение о том, что размер принимающего поля меньше размера посылаемого.

Примеры пересылки литералов и фигуральных констант:

Посылаемое поле	Принимающее поле	
	шаблон	результат пересылки
849.5	9 (4) Т9 (2)	0849∧50
НУЛЬ	99Т999	00000
ПРОБЕЛ	А (5)	␣␣␣␣␣
'А 2—БУКВЕННОГО КЛАССА'	Х (15)	А2—БУКВЕННОГО—К*
КАВЫЧКА	Х (3)	'''
ВСЕ '_'	Х (8)	_ _ _ _ _ _ _ _

При пересылке группового данного в групповое никакого приведения в соответствие с шаблонами элементарных данных принимающего поля не производится, поэтому описания элементарных данных в групповых должны полностью совпадать. Если они не совпадают, при трансляции выдается предупреждение. Если при этом размер посылаемого поля превышает размер принимающего поля, то пересылаться будет лишь часть, помещающаяся в принимающем поле.

Примеры.

Пусть имеем следующие описания групповых данных А1, Б1, А2, Б2, А3, Б3:

```
01      А1.
    02      А11  ШАБЛОН    99Т9.
    02      А21  ШАБЛОН    Х(3) ПОВТОРЯЕТСЯ 2 РАЗА.
    02      А31  ШАБЛОН    99 ДЛЯ ВЫЧИСЛЕНИЙ.
01      А2.
    02      А12  ШАБЛОН    99Т9.
    02      А22  ШАБЛОН    Х(3).
    02      А32  ШАБЛОН    ПП99.9.
    02      А42  ШАБЛОН    9(3).
01      А3.
    02      А13  ШАБЛОН    9(3).
    02      А23  ШАБЛОН    Х(17).
    02      А33  ШАБЛОН    А(10).
01      Б1.
    02      Б11  ШАБЛОН    99Т9.
    02      Б21  ШАБЛОН    Х(3) ПОВТОРЯЕТСЯ 2 РАЗА.
    02      Б31  ШАБЛОН    99 ДЛЯ ВЫЧИСЛЕНИЙ.
01      Б2.
    02      Б12  ШАБЛОН    9Т99 ДЛЯ ВЫВОДА.
    02      Б22  ШАБЛОН    Х(7).
    02      Б32  ШАБЛОН    М(3)99.
    02      Б42  ШАБЛОН    +++9.
01      Б3.
    02      Б13  ШАБЛОН    П(3)9.
    02      Б23  ШАБЛОН    Х(13).
    02      Б33  ШАБЛОН    А(5).
```

Оператор ПОМЕСТИТЬ А1 В Б1 является правильным: полностью совпадают описания пунктов.

* См. примечание на стр. 246.

Оператор ПОМЕСТИТЬ А2 В Б2 является неправильным, так как описания элементарных данных не совпадают. При трансляции будет выдано соответствующее предупреждение.

Оператор ПОМЕСТИТЬ А3 В Б3 тоже неправильный. При трансляции будет выдано сообщение о несовпадении описаний и о том, что размер посылаемого данного больше принимающего. Перешлется только часть данного А3, помещающаяся в Б3.

Оператор ПРОСМОТРЕТЬ дает возможность подсчитать число появлений заданной литеры в элементарной величине или определить местоположение литеры в значении данного. Кроме того, он позволяет заменить некоторую заданную литеру другой заданной литерой.

Формат оператора:

ПРОСМОТРЕТЬ название-данного

```
{ СЧИТАЯ { ДО ПЕРВОГО } нечисловой-литерал-1 [ЗАМЕНЯЯ НА нечисловой-литерал 2] }
                { ВСЕ }
{ ЗАМЕНЯЯ       { ВЕДУЩИЕ }  нечисловой-литерал-3 [НА нечисловой-литерал-4] }
                { ДО ПЕРВОГО }
                { ПЕРВЫЙ }
```

Все нечисловые литералы в формате должны состоять из одной литеры.

Просмотр данного производится посимвольно слева направо.

Используя вариант со словом СЧИТАЯ, можно произвести подсчет:

а) общего количества литер до первого появления литеры-1 (вариант ДО ПЕРВОГО);

б) общего количества появлений литеры-1 в данном (вариант ВСЕ);

в) количества появлений литеры-1 до первой отличной от нее литеры (вариант ВЕДУЩИЕ).

Результат подсчета всегда помещается в специальное поле СЧЕТЧИК, которое рассматривается как данное с шаблоном 9(5) и может быть использовано при дальнейшей обработке. Данное СЧЕТЧИК не должно описываться в разделе данных. Слово СЧЕТЧИК относится к зарезервированным словам.

Используя вариант формата со словом ЗАМЕНЯЯ, можно заменить во всех указанных вариантах одну литеру другой. Значение просматриваемого поля тем самым изменяется.

Просматриваемое данное не должно быть цифровым.

Примеры.

1. Пусть данное А1 имет значение ВЫСКАЗЫВАНИЕ

Вариант	Значение А1 после выполнения оператора	Значение счетчика
СЧИТАЯ ВСЕ 'В'	ВЫСКАЗЫВАНИЕ	2
СЧИТАЯ ВЕДУЩИЕ 'В'	ВЫСКАЗЫВАНИЕ	1
СЧИТАЯ ДО ПЕРВОГО 'З'	ВЫСКАЗЫВАНИЕ	5

Вариант	Значение А1 после выполнения оператора	Значение счетчика
ЗАМЕНЯЯ ВСЕ 'Е' на 'Я'	ВЫСКАЗЫВАНИЯ	Сохраняется прежнее
ЗАМЕНЯЯ ВЕДУЩИЕ 'В' НА 'А'	АЫСКАЗЫВАНИЕ	Сохраняется прежнее
ЗАМЕНЯЯ ДО ПЕРВОГО 'З' НА 'Я' СЧИТАЯ ВСЕ 'А'	ЯЯЯЯЗЫВАНИЕ	Сохраняется прежнее
ЗАМЕНЯЯ НА 'Л'	ВЫСКЛЗЫВАНИЕ	2

2. БЛ0. ПРОСМОТРЕТЬ ШНП СЧИТАЯ ВСЕ '1' ЗАМЕНЯЯ НА'—'.
ЕСЛИ СЧЕТЧИК НЕ=0 ПЕРЕЙТИ К БЛ1.
ПРОСМОТРЕТЬ ШНП СЧИТАЯ ВСЕ '2' ЗАМЕНЯЯ НА '—'.
ЕСЛИ СЧЕТЧИК НЕ =0 ПЕРЕЙТИ К БЛ2.
ПРОСМОТРЕТЬ ШНП СЧИТАЯ ВСЕ '3' ЗАМЕНЯЯ НА '—'.
ЕСЛИ СЧЕТЧИК НЕ=0 ПЕРЕЙТИ К БЛ3.
ПЕРЕЙТИ К БЛ4.
БЛ1. ВЫЧИСЛИТЬ НАЛОГ=5.5*ОБЛАГАЕМАЯ — 120,
ПЕРЕЙТИ К БЛ0.
БЛ2. ВЫЧИСЛИТЬ НАЛОГ=.6*ОБЛАГАЕМАЯ, ПЕРЕЙТИ К БЛ0.
БЛ3. ВЫЧИСЛИТЬ НАЛОГ=0,38*ОБЛАГАЕМАЯ — 70.
БЛ4. ...

Здесь данное ШНП имеет шаблон X(3) и может содержать любую комбинацию из четырех символов:—, 1, 2, 3, где 1, 2 и 3 — значения шифра налогоплательщика. Необходимо произвести расчет данного НАЛОГ по всем заданным в ШНП шифрам налогоплательщика.

Оператор УСТАНОВИТЬ служит для установки значения названия-индекса, связанного с элементами некоторого списка (таблицы).

Формат оператора:

$$\text{УСТАНОВИТЬ} \left\{ \begin{array}{l} \text{название-индекса-1} \\ \text{название-данного-1} \end{array} \right\} \text{НА} \left\{ \begin{array}{l} \text{название-индекса-2} \\ \text{название-данного-2} \\ \text{числовой-литерал-2} \end{array} \right\}$$

Названия данных должны быть цифровыми элементарными данными, не содержащими в описании неявной десятичной точки. Названия данных не должны индексироваться. Если в формате использован числовой литерал, то он должен быть целым числом.

Названия индексов должны быть определены своим появлением во фразе ИНДЕКСИРУЕТСЯ при описании некоторой таблицы (списка) в разделе данных.

Если устанавливается название-индекса-1, номер повторения элемента, соответствующий названию-индекса-2 или задаваемый названием-данного-2 (числовым-литералом-2), преобразуется к виду, используемому для определения адреса элемента таблицы, связанной с названием-индекса-1, и помещается в название-индекса-1.

Название-данного-1, являющееся индексом, может быть установлено только на значение названия-индекса-2 (для времен-

ного запоминания названия-индекса-2) или значение другого индекса название-данного-2. Пересылка производится без каких-либо преобразований.

Название-данного-1, не являющееся индексом, может быть установлено на название-индекса-2. При этом название-индекса-2 будет преобразовано в номер повторения элемента, соответствующий значению названия-индекса-2, который и будет помещен в название-данного-1. Все остальные сочетания операндов не допускаются.

Пример.

Рассмотрим приведенный ранее пример умножения матрицы на вектор. Пусть данный и результирующий векторы и матрица описаны следующим образом:

```
01 ВЕКТОР.
02 Д Ш 9 ПОВТОРЯЕТСЯ 3 РАЗА.
01 РЕЗУЛЬТАТ.
02 Ф Ш 9 ПОВТОРЯЕТСЯ 3 РАЗА.
01 МАТРИЦА.
   02 СТРОКА ПОВТОРЯЕТСЯ 3 РАЗА ИНДЕКСИРУЕТСЯ ИНД1.
      03 Е Ш 9 ПОВТОРЯЕТСЯ 3 РАЗА ИНДЕКСИРУЕТСЯ ИНД2.
```

Пусть ИНД3 и ИНД4 — индексы, описанные в разделе данных. При таком задании индексов поставленная в приведенном ранее примере задача решается с помощью той же группы операторов.

Операторы ввода-вывода. К этой группе относятся операторы ОТКРЫТЬ, ЧИТАТЬ, ПИСАТЬ, ЗАКРЫТЬ, ПРИНЯТЬ, ВЫДАТЬ. Они служат для ввода данных с внешних носителей в оперативную память и вывода из оперативной памяти на внешние носители.

Оператор ОТКРЫТЬ подготавливает к обработке входные и выходные массивы: производит проверку (если массив входной) или запись (если массив выходной) начальной метки, проверяет правильность использования внешинх устройств и зон памяти для массивов и т. п.

Формат оператора:

ОТКРЫТЬ { ВХОДНОЙ / ВЫХОДНОЙ } название-массива-1 [БЕЗ ПЕРЕМОТКИ]

[название-массива-2 [БЕЗ ПЕРЕМОТКИ]...] [, { ВХОДНОЙ / ВЫХОДНОЙ }

название-массива-3 [БЕЗ ПЕРЕМОТКИ] [,название-массива-4 [БЕЗ ПЕРЕМОТКИ]...] ...]

Выполнение оператора ОТКРЫТЬ для массива должно предшествовать выполнению операторов ЧИТАТЬ (ПИСАТЬ) и ЗАКРЫТЬ для этого массива, так как не открытый массив недоступен для обработки. В пределах одной программы массив может быть повторно открыт лишь при условии, что он был предварительно закрыт. Это происходит потому, что глагол ОТКРЫТЬ заносит в таблицу состояния массива рабочей про-

граммы, находящейся в оперативной памяти машины, признак того, что массив открыт. В процессе выполнения рабочей программы состояние этого признака контролируется. Значение признака остается неизменным до тех пор, пока оно не будет изменено действием глагола ЗАКРЫТЬ или пока программа не будет удалена из оперативной памяти.

Вариант БЕЗ ПЕРЕМОТКИ используется только для массивов на магнитной ленте. Если использован вариант БЕЗ ПЕРЕМОТКИ, то перед открытием массива не происходит перемотки ленты в начало. Имеет смысл использовать эту фразу при открытии очередного массива, находящегося на одной ленте с другими, для поиска массива с движением ленты вперед. Открытие массива, предшествующего на ленте обрабатываемому, должно производиться с перемоткой, т. е. с движением ленты в обратном направлении. Первый массив на ленте с несколькими массивами или единственный массив на ленте должен также открываться всегда с перемоткой.

Оператор ЧИТАТЬ помещает очередную запись указанного массива в зону входной записи (делает доступной обработке). При этом он выполняет две функции:

1) считывает порцию документов с внешнего носителя в зону ввода-вывода массива и подает первый документ из зоны ввода-вывода в зону записи;

2) последовательно считывает документы из зоны ввода-вывода в зону записи до тех пор, пока не исчерпает ее полностью. Когда считан последний документ из зоны ввода-вывода и глагол ЧИТАТЬ обращается к пустой зоне ввода-вывода, он выполняет свою первую функцию.

Формат оператора:

ЧИТАТЬ название-массива В КОНЦЕ повелительный-оператор.

Массив предварительно должен быть открыт.

Для каждого массива отводится в оперативной памяти только одна зона записи. Поэтому, если в массиве есть записи различных типов, все они будут подаваться в одну и ту же зону, отведенную в расчете на максимальную по размеру запись. Для определения того, какого типа запись поступила в зону записи, используется проверка типа записи.

Если при попытке прочесть очередную запись массива обнаруживаются конечные метки массива (что свидетельствует о том, что все записи массива исчерпаны), выполняется оператор, указанный после служебных слов В КОНЦЕ. Читать массив после этого в той же программе, не закрыв и не открыв его вновь, нельзя.

Пример.

ЧИТАТЬ ОПЕРАТИВНЫЙ В КОНЦЕ ПЕРЕЙТИ К ФИНИШ.

Оператор ПИСАТЬ используется для переноса очередной записи из оперативной памяти в выходной массив.

Формат оператора:

ПИСАТЬ название-записи [{СПЕРВА / ЗАТЕМ} ПРОДВИГАЯ целое {СТРОК / СТРОКИ / СТРОКУ}]

Выполнению этого оператора должно предшествовать выполнение оператора ОТКРЫТЬ ВЫХОДНОЙ для массива, к которому относится запись, указанная в операторе. Запись из выходного массива недоступна до тех пор, пока этот массив не будет закрыт и вновь открыт как входной.

Оператор ПИСАТЬ также выполняет одновременно две функции:

1) последовательно переносит документы из зоны записи в зону ввода-вывода выходного массива до полного заполнения последней;

2) как только зона ввода-вывода заполнена, производит выгрузку ее на внешний носитель.

При выдаче выходного массива на печать удобно отделять одну запись от другой пустыми строками. С этой целью при выдаче записи на печать используется вариант формата глагола ПИСАТЬ со словом ПРОДВИГАЯ, который и позволяет продвинуть бумагу до (СПЕРВА) или после (ЗАТЕМ) печати записи на указанное количество строк.

Если «целое» равно К, то будет пропущена К—1 пустая строка перед или после печати указанной записи. К, равное 1, равносильно тому, что фраза ПРОДВИГАЯ вообще отсутствует, т. е. очередная запись будет печататься на следующей строке.

В описании записи, предназначенной для вывода на печать с помощью глагола ПИСАТЬ, необходимо предусмотреть наличие пробелов между данными, соблюдение геометрии выходного документа и т. д. С этой целью в записи обычно вводят пробелы, разделители граф и прочие заполнители.

Пример.

Пусть документы некоторого массива требуется распечатать по следующей схеме:

Тогда запись, предназначенную для вывода на печать с помощью глагола ПИСАТЬ, можно описать следующим образом:

```
01 ЗАП1.
02 ЗАП III X(12) ЗНАЧ'_____1_'.
02 А III 9(3).
02 ЗАП Ш X(3) ЗНАЧ '_1_'.
02 Б III X(5).
02 ЗАП Ш X(4) ЗНАЧ '_1__'.
02 Е Ш 99.
02 ЗАП Ш X(4) ЗНАЧ'__1_'.
02 Ф Ш X(8).
02 ЗАП Ш XX ЗНАЧ'_1'.
```

Результатом выполнения оператора

ПИСАТЬ ЗАПО1 ЗАТЕМ ПРОДВИГАЯ 2 СТРОКИ.

будет документ ЗАПО1, распечатанный по указанной схеме, и одна строка пробелов после него.

Оператор ЗАКРЫТЬ выполняет необходимые операции по завершению обработки массива, например, запись (если массив выходной) или проверку конечных меток входного массива, освобождение устройств и пр.

Оператор ЗАКРЫТЬ изменяет значение признака в таблице состояния массива рабочей программы.

Формат оператора:

ЗАКРЫТЬ название-массива-1 [С ОСВОБОЖДЕНИЕМ] [, название-массива-2 [С ОСВОБОЖДЕНИЕМ] ...]

Массив сначала должен быть открыт. Каждому ЗАКРЫТЬ для массива должно соответствовать свое ОТКРЫТЬ. Рабочие зоны (зона записи и зона ввода-вывода) массива при закрытии освобождаются и могут быть использованы другими массивами, имеющими общую зону с данным.

Если закрывается входной массив, не обработанный до конца, проверка конечной метки не производится. Вообще говоря, входной массив, который повторно не открывается в одной и той же программе, можно не закрывать.

При закрытии массива С ОСВОБОЖДЕНИЕМ устройство, занятое этим массивом, освобождается. Магнитная лента сматывается для снятия.

Если на одну катушку нужно записать несколько массивов, все массивы, кроме последнего, следует записывать без освобождения (а открывать все массивы без перемотки).

Пример.

Пусть имеется массив КАТАЛОГ с документами следующей структуры:

```
01 КАРТА.
02 НАЗВАНИЕ Ш Х(20).
02 ДАТА1 Ш 9(4).
02 АВТОР1 Ш Х(40).
02 ШИФР1 Ш 9(5).
02 ПОДРАЗДЕЛ1 Ш 99.
```

Документы исходного массива необходимо перекомпоновать в документы вида

```
01 ЕДИНИЦА.
02 ШИФР2 Ш 9(5).
02 ПОДРАЗДЕЛ2 Ш 99.
```

и записать их в массив МАЛЫЙ-КАТАЛОГ. Это достигается путем использования следующей группы операторов:

```
НАЧ. ОТКРЫТЬ ВХОДНОЙ КАТАЛОГ. ОТКРЫТЬ ВЫХОДНОЙ МАЛЫЙ-КАТАЛОГ.
ЧИТ. ЧИТАТЬ КАТАЛОГ В КОНЦЕ ПЕРЕЙТИ К КОН.
     ПОМЕСТИТЬ ШИФР1 В ШИФР2.
     ПОМЕСТИТЬ ПОДРАЗДЕЛ1 В ПОДРАЗДЕЛ2.
     ПИСАТЬ ЕДИНИЦА. ПЕРЕЙТИ К ЧИТ.
КОН. ЗАКРЫТЬ КАТАЛОГ, МАЛЫЙ-КАТАЛОГ.
     ОСТАНОВИТЬ РАБОТУ.
```

Оператор ПРИНЯТЬ используется для ввода данных небольшого объема с пишущей машинки.

Формат оператора:

ПРИНЯТЬ название-данного

Принимаемое данное должно быть обязательно описано в разделе данных исходной программы как элементарное нецифровое данное. Размер принимаемого данного не должен превышать 92 литер.

Поскольку стандартная программа, реализующая оператор ПРИНЯТЬ, относится к загружаемым программам (т. е. сохраняется в оперативной памяти только на время действия ПРИНЯТЬ), имеет смысл избегать частого использования этого оператора. В противном случае будет затрачиваться значительное машинное время на повторную загрузку программы с магнитной ленты для каждого нового ПРИНЯТЬ. Чтобы сократить число обращений к ленте, предпочтительнее принимать сложные данные и делить их на составные части программным путем, если это возможно.

Пример.

ПРИНЯТЬ ГОДМЕС. ПОМЕСТИТЬ ГОДМЕС В ГОДМЕСЯЦ. ПОМЕСТИТЬ ГОДМЕСЯЦ В МЕСЯЦ. ВЫЧИСЛИТЬ ГОД=ГОДМЕС/100.

Здесь данное ГОДМЕС имеет шаблон X(6), данное ГОДМЕСЯЦ — 9(6), данное МЕСЯЦ — 99, данное ГОД — 9(4).

Набор требуемой информации на пишущей машинке оператор ЭВМ должен начинать после печати названия данного, указанного в формате. Ответ оператора имеет вид: — ААААА; ББ...Б◊, где ААААА — имя задания, ББ...Б — принимаемое данное.

Оператор ВЫДАТЬ предназначен для выдачи данных небольшого объема на устройство печати.

Формат оператора:

ВЫДАТЬ { числовой-литерал-1 / название-данного-1 / нечисловой-литерал-1 / фигуральная-константа-1 } [, { числовой-литерал-2 / название-данного-2 / нечисловой-литерал-2 / фигуральная-константа-2 } ...] [НА условное-название-1].

Выдаваемый числовой литерал не должен иметь знака. При необходимости выдать число со знаком можно выдать нечисловой литерал.

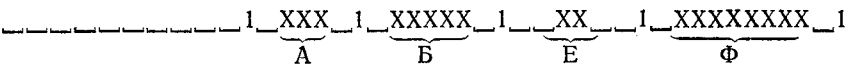
Условное-название-1 в формате есть условное название, присвоенное устройству (ПЧ1), используемому для вывода на него данных, в параграфе СПЕЦИАЛЬНЫЕ-НАЗВАНИЯ раздела оборудования. Если условное-название-1 отсутствует, предполагается ПЧ1.

Название специального данного — СЧЕТЧИК, как видно из формата, не может непосредственно использоваться в операторе ВЫДАТЬ. Для выдачи содержимого СЧЕТЧИКА необходимо предварительно переслать его в какое-нибудь рабочее поле, а затем выдать это рабочее поле.

В качестве фигуральной константы могут быть использованы НУЛЬ, ПРОБЕЛ, КАВЫЧКА. В этом случае выдается только одна соответствующая литера.

Порядок выдачи величин соответствует порядку перечисления их в операторе. Все величины, описанные с помощью одного ВЫДАТЬ, выдаются в одну строку устройства без каких-либо промежутков между ними. Суммарный размер выдаваемых данных должен быть не более 128 литер.

Пример.
Пусть данные записи
01 ЗАП1.
02 А Ш 9(3).
02 Б Ш 9(5).
02 Е Ш XX.
02 Ф Ш X(8).
требуется напечатать в следующем виде:



С помощью оператора ВЫДАТЬ это выполняется следующим образом:
ВЫДАТЬ СТРОКА-ПРОБЕЛОВ, ЕДИНИЦА, А, ЕДИНИЦА, Б, ЕДИНИЦА, '_', Е, '_', ЕДИНИЦА, Ф, '_1'.
Здесь данное СТРОКА-ПРОБЕЛОВ имеет шаблон X(9) и значение ВСЕ '_'; данное ЕДИНИЦА имеет шаблон X(3) и значение '_1_'.

Оператор упорядочения массива СОРТИРОВАТЬ служит для упорядочения записей массива по указанным ключам.

Формат оператора:

СОРТИРОВАТЬ название-массива-1 ПО {ВОЗРАСТАНИЮ / УБЫВАНИЮ} КЛЮЧА
название-данного-1 [, название-данного-2 ...]
[, ПО {ВОЗРАСТАНИЮ / УБЫВАНИЮ} КЛЮЧА название-данного-3 [, название-данного-4 ...] ...] [ПОЛУЧАЯ название-массива-2]]

Массив-1 должен быть описан в разделе данных на уровне ОС. Каждое название данного должно представлять собой название элементарного пункта записи этого массива. Массив-2 должен быть описан в статье описания массива с индикатором ОМ. Метод записи массива-2 должен быть таким же, как и в массиве-1. Кроме того, в массиве-1 и массиве-2 должны совпадать описания и порядок элементарных данных. Размер блока массива-2 должен быть таким, чтобы в области, зарезервированной для сортировки, помещалось целое число блоков массива-2 или чтобы он не превосходил $^5/_6$ этой области.

Следующие за словом КЛЮЧА названия ключей сортировки перечисляются в порядке убывания их старшинства. Описания пунктов данных, являющихся ключами сортировки, не могут ни содержать фразу ПОВТОРЯЕТСЯ, ни подчиняться ста-

тьям, содержащим такую фразу. Если в сортируемом массиве несколько типов записей, они должны быть одинакового размера и ключи сортировки должны принадлежать одному типу.

Массив-1 и массив-2 не нужно открывать перед началом сортировки (и, следовательно, закрывать в конце). Во время сортировки никаких других открытых массивов не должно быть. Для массива-1 и массива-2 в разделе оборудования могут быть предназначены только магнитные ленты. При этом сортируемому массиву предназначается две магнитные ленты.

Пример.

Пусть имеется массив СПРАВОЧНЫЙ с документами следующей структуры:

```
01 ЗАПО1.
02 ЦЕХ Ш 9(3).
02 ТАБНОМ Ш 9(6).
02 УЧАСТОК Ш 9(3).
02 ШНП Ш 9.
02 КАТЕГОРИЯ Ш 99.
02 ФИО Ш Х(20).
```

Необходимо упорядочить этот массив по возрастанию реквизита ЦЕХ, внутри цеха — по возрастанию реквизита УЧАСТОК, внутри участка — по возрастанию реквизита ТАБНОМ. Имя рассортированного массива — СПРАВСОРТ.

Тогда оператор имеет вид:

СОРТИРОВАТЬ СПРАВОЧНЫЙ ПО ВОЗРАСТАНИЮ КЛЮЧА ЦЕХ, УЧАСТОК, ТАБНОМ ПОЛУЧАЯ СПРАВСОРТ.

Операторы связи между программами. В эту группу входят операторы ВЫЗВАТЬ, ВЫЙТИ ИЗ-ПРОГРАММЫ, ОСВОБОДИТЬ, ВОЙТИ.

Оператор ВЫЗВАТЬ используется для вызова программы библиотеки программ на языке загрузки и передачи ей управления.

Формат оператора:

ВЫЗВАТЬ название-программы [ИСПОЛЬЗУЯ нечисловой-литерал-1, название-данного-1 [, название-данного-2 ...] [, нечисловой-литерал-2, название-данного-3 [, название-данного-4 ...] ...].

Название-программы — это название вызываемой программы, которое составляется по правилам образования слов языка, но содержит не более 5 литер. В одной и той же исходной программе названия программ в операторах ВЫЗВАТЬ и ВОЙТИ не должны совпадать.

Вариант ИСПОЛЬЗУЯ применяется в случае, если вызывающая программа передает данные вызываемой программе через общие области или через параметры. Названия данных во фразе ИСПОЛЬЗУЯ есть названия данных, описанных в секции массивов, секции рабочей памяти вызывающей программы или (если эта программа является в то же время и вызываемой) в секции связи на уровнях 01 или 77.

Данные, появляющиеся во фразе ИСПОЛЬЗУЯ, становятся доступными вызываемой программе. Если данные требуется передать через параметры сообщения, применяется вариант

ИСПОЛЬЗУЯ название-данного-1 [, название-данного-2...],

если через общие области — вариант

ИСПОЛЬЗУЯ нечисловой-литерал-1, название-данного-1 [, название-данного-2...].

Нечисловой-литерал-1 определяет имя общей области и должен состоять не более чем из пяти литер. В общую область с именем нечисловой-литерал-1 входят данные, перечисленные вслед за нечисловым литералом. В списке данных общей области название записи, передаваемой из секции массивов, может принадлежать только массиву с одним типом записи, не имеющему общей зоны с другими массивами. Нельзя использовать в списке данных общей области название данного, описанного с фразой ПЕРЕОПРЕДЕЛЯЕТ. Если в нескольких ВЫЗВАТЬ встречаются одинаковые нечисловые литералы, то следующие за ними списки данных должны совпадать по количеству, названию и порядку перечисления. Одно и то же данное может принадлежать разным областям. Если данные передаются вызываемой программе через параметры, то как групповые, так и элементарные данные не могут быть названы повторно в одном и том же списке данных. В рабочей программе параметрами обращения к вызываемой программе будут адреса данных, названных в списке данных оператора ВЫЗВАТЬ, причем адреса данных, имеющих в списке нечетные номера, будут располагаться по второму адресу, а имеющие четные номера — по первому адресу последовательно в ячейках, следующих за командой «идти к программе».

Пример.

Пусть в программе ПРОГР имеется оператор ВЫЗВАТЬ ПРОГ1 ИСПОЛЬЗУЯ 'ОБЩ1', ДАННОЕ1, ДАННОЕ2, 'ОБЩ2', ДАННОЕ3,

где ПРОГ1 — имя вызываемой программы, написанной на языке КОБОЛ, а ДАННОЕ1, ДАННОЕ2 и ДАННОЕ3 — данные из секции массивов и секции рабочей памяти ПРОГР. Тогда эти же данные в программе ПРОГ1 должны быть описаны в секции связи раздела данных. Раздел процедур программы ПРОГ1 должен начинаться с заголовка РАЗДЕЛ ПРОЦЕДУР ИСПОЛЬЗУЯ 'ОБЩ1', ДАННОЕ1, ДАННОЕ2, 'ОБЩ2', ДАННОЕ3.

Если вызываемая программа написана на ЯСК, в качестве параметра может передаваться любое название данного, в том числе и название массива.

Если в качестве параметра использовано название массива, параметром в обращении будет адрес А таблицы информации для этого массива. Увеличенный на 6 адрес А даст адрес Д описания массива.

Если данные передаются и через параметры, и через общие области, в начале в варианте ИСПОЛЬЗУЯ должен находиться список параметров, а затем списки данных общих областей.

Количество общих областей в программе не должно быть больше 27, включая те, которые образуются из списков данных в заголовке раздела процедур.

Вызываемая программа загружается в оперативную память в рабочее время в момент первого обращения к ней. При повторном обращении загрузка производится только в случае, если память программы освобождалась.

Оператор ВЫЙТИ ИЗ-ПРОГРАММЫ возвращает управление вызывающей программе. Если оператор используется в головной программе, управление передается ДИСПЕТЧЕРу.

Формат оператора:

ВЫЙТИ ИЗ-ПРОГРАММЫ.

Оператор ВЫЙТИ ИЗ-ПРОГРАММЫ всегда должен быть оформлен в виде отдельного параграфа, в котором он должен быть единственным оператором.

Оператор не должен быть обязательно последним в разделе процедур. С помощью этого оператора управление передается в вызывающую программу следующему за ВЫЗВАТЬ оператору. При этом память, занимаемая программой, не освобождается и при повторном обращении к этой программе с помощью ВЫЗВАТЬ загрузки вызываемой программы не происходит.

Оператор ОСВОБОДИТЬ используется для освобождения участка памяти, занятого вызванной программой.

Формат оператора:

ОСВОБОДИТЬ.

Оператор ОСВОБОДИТЬ прерывает связи между программами (вызывающей и последней загруженной программой). Если для некоторой программы выполнилось ОСВОБОДИТЬ, то при повторном обращении к ней с помощью ВЫЗВАТЬ вызываемая программа будет вновь загружаться, прежде чем ей будет передано управление.

Использование оператора ОСВОБОДИТЬ позволяет загружать последовательные сегменты задачи на одно и то же место в памяти.

Оператор ВОЙТИ позволяет передавать управление библиотечной программе, предварительно собранной с данной программой.

Формат оператора:

ВОЙТИ в название-программы [ИСПОЛЬЗУЯ нечисловой-литерал-1, название-данного-1 [, название-данного-2, ...] [, нечисловой-литерал-2, название-данного-3 [, название-данного-4, ...] ...]]

Вариант с фразой ИСПОЛЬЗУЯ применяется, если внешней программе необходимо передать данные внутренней программы через общие области или параметры, и подчиняется тем же правилам, что и вариант ИСПОЛЬЗУЯ в операторе ВЫЗВАТЬ. Для внутренних программ, написанных на ЯСК, допустим также следующий формат оператора:

ВОЙТИ в название-программы [(название-данного-1 | числовой-литерал-1 | нечисловой-литерал-1) [, (название-данного-2 | числовой-литерал-2 | нечисловой-литерал-2) ...]]

Название программы должно быть образовано по правилам образования слов в КОБОЛе и состоять не более чем из 5 литер.

За названием программы в круглых скобках перечисляются, если необходимо, параметры программы, которые могут быть представлены названиями данных, описанными в разделе данных, и (или) литералами. Названия данных не должны индексироваться.

Транслятор преобразует эти параметры в адреса данных или литералов, этими адресами будет оперировать внутренняя программа, которой передается управление по глаголу ВОЙТИ.

Порядок следования параметров во внутренней программе определяется порядком перечисления их в операторе ВОЙТИ. При этом если внутренняя программа написана на ЯСК, то ее начало должно быть оформлено следующим образом:

```
        БАЗ     0
        РЗВ     3
А       РЗВ     1
Б       РЗВ     1
.  .  .  .  .  .  .
```

Здесь А, Б, ...— ячейки, резервируемые для адресов передаваемых параметров. Адреса передаваемых параметров располагаются по два в одной ячейке в следующем порядке:

```
a2      a1
a4      a3
...     ...
```

где a1, a2, a3, ...— адреса параметров в операторе ВОЙТИ в название-программы (a1, a2, a3, a4, ...).

В случае, если внутренняя программа написана на КОБОЛе, передаваемые параметры должны быть описаны в секции связи этой внутренней программы.

4.6. ФОРМАТЫ ВХОДНОГО ЯЗЫКА

Все форматы синтаксиса языка КОБОЛ сведены в единую таблицу (табл 4.4). В приводимой таблице форматов используются следующие сокращения:

н-у — название-устройства
у-н — условное-название
н-м — название-массива
н-л — нечисловой-литерал
н-д — название-данного

н-и — название-индекса
ч-л — числовой-литерал
ф-к — фигуральная константа
н-п — название-процедуры

Таблица 4.4

Формат раздела идентификации

РАЗДЕЛ ИДЕНТИФИКАЦИИ.
ПРОГРАММА. комментирующее-предложение.
[АВТОР. комментирующее-предложение.]
[ДАТА-НАПИСАНИЯ. комментирующее-предложение.]
[ЗАМЕЧАНИЯ. комментирующее-предложение.]

Формат раздела оборудования

РАЗДЕЛ ОБОРУДОВАНИЯ.
СЕКЦИЯ КОНФИГУРАЦИИ.
[ИСХОДНАЯ МАШИНА. МИНСК-32.]
РАБОЧАЯ МАШИНА. МИНСК-32 [; РАЗМЕР ПАМЯТИ целое ЛИСТОВ]
[СПЕЦИАЛЬНЫЕ НАЗВАНИЯ. н-у-1 ЕСТЬ у-н-1. [н-у-2 ЕСТЬ у-н-2 . . .]]
[СЕКЦИЯ ВВОДА-ВЫВОДА.
УПРАВЛЕНИЕ МАССИВАМИ.
ДЛЯ н-м-1 ПРЕДНАЗНАЧИТЬ н-у-1 [, н-у-2] [; СО СМЕНОЙ КАТУШЕК] [; РЕЗЕРВИРОВАТЬ целое ДОПОЛНИТЕЛЬНЫХ ЗОН].
[УПРАВЛЕНИЕ-ВВОДОМ-ВЫВОДОМ.
[ПЕРЕПРОГОН [НА н-у] ПОСЛЕ { КОНЦА КАТУШКИ / целое ЗАПИСЕЙ } н-м-1]
[ОБЩАЯ ЗОНА ДЛЯ н-м-2, н-м-3 [, н-м-4 . . .] . . .] [НА ОДНОЙ КАТУШКЕ н-м-5, н-м-6 [, н-м-7 . . .] . . .]]]

Формат раздела данных

Структура раздела данных

РАЗДЕЛ ДАННЫХ.
[СЕКЦИЯ МАССИВОВ.
Статья описания массива-1
Статья описания записи 1-го типа массива-1.
. . . . . .
Статья описания записи *к*-го типа массива-1.
Статья описания массива-2.
Статья описания записи 1-го типа массива-2.
. . . . . .
. . . . . .]
[СЕКЦИЯ РАБОЧЕЙ-ПАМЯТИ.
Статьи описания несвязанных рабочих полей или констант.
Статьи описания связанных рабочих полей.]
[СЕКЦИЯ СВЯЗИ.
Статьи описания несвязанных полей данных.
Статьи описания связанных полей данных]

Формат раздела данных

Формат статьи описания массива

{ ОМ / ОС } н-м [; МЕТОД { СТАНДАРТ / ПЛОТНЫЙ / С-РАЗДЕЛИТЕЛЯМИ н-л }]
[; В БЛОКЕ целое ЗАПИСЕЙ]; МЕТКИ

{ СТАНДАРТНЫ, ШИФР МАССИВА н-л [,ПЕРИОД ГОДНОСТИ целое-1] / ОПУЩЕНЫ }

Формат статьи описания записи

номер-уровня { н-д / ЗАПОЛНИТЕЛЬ / ЗАП } [; ПЕРЕОПРЕДЕЛЯЕТ н-д-1]

[; { ШАБЛОН / Ш } строка-литер] [; { ЗНАЧЕНИЕ / ЗНАЧ } { литерал / ф-к }]

[; ПОВТОРЯЕТСЯ целое { РАЗ / РАЗА }] [; ИНДЕКСИРУЕТСЯ н-и]

[; ДЛЯ { ВЫВОДА / ВЫЧИСЛЕНИЙ / ВЫЧ }] [; ПРОБЕЛ КОГДА НУЛЬ]

Форматы раздела процедур

Формат заголовка

РАЗДЕЛ ПРОЦЕДУР [ИСПОЛЬЗУЯ н-л-1, н-д-1 [, н-д-2 ...]
[, [н-л-2] н-д-3 [, н-д-4 ...] ...]]

Арифметический глагол

ВЫЧИСЛИТЬ н-д-1 = { н-д-2 / ч-л / арифметическое выражение }

[ОКРУГЛЯЯ] [ПРИ ПЕРЕПОЛНЕНИИ повелительный-оператор]

Глаголы управления последовательностью процедур

ПЕРЕЙТИ К н-п-1 [, н-п-2 [, н-п-3 ...] В ЗАВИСИМОСТИ ОТ н-д]

ИЗМЕНИТЬ н-п-1 ДЛЯ ПЕРЕХОДА К н-п-2

ВЫПОЛНИТЬ н-п-1 [ПО н-п-2] { целое / н-д } { РАЗ / РАЗА }

ВЫПОЛНИТЬ н-п-1 [ПО н-п-2] ДО условие

ВЫПОЛНИТЬ н-п-1 [ПО н-п-2] МЕНЯЯ { н-и-1 / н-д-1 } ОТ { н-и-2 / н-д-2 / ч-л-2 }

НА { ч-л-3 / н-д-3 } ДО условие-1 [МЕНЯЯ { н-и-4 / н-д-5 } ОТ { н-и-5 / ч-л-5 / н-д-5 }

Форматы раздела процедур

Глаголы управления последовательностью процедур

НА { ч-л-6 / н-д-6 } ДО условие-2]

ВЫЙТИ.

ЕСЛИ условие { повелительный-оператор-1 / ; СЛЕДУЮЩЕЕ ПРЕДЛОЖЕНИЕ } [; ИНАЧЕ { повелительный-оператор-2 / СЛЕДУЮЩЕЕ ПРЕДЛОЖЕНИЕ }]

На месте условия в формате может быть:

1. { н-и-1 / н-д-1 / литерал-1 / арифметическое-выражение-1 } [НЕ] { > БОЛЬШЕ / < МЕНЬШЕ / = РАВНО } { н-и-2 / н-д-2 / литерал-2 / арифметическое-выражение-2 }

2. н-д [НЕ] { ЦИФРОВОЕ / БУКВЕННОЕ }

3. ЗАПИСЬ название-записи

ОСТАНОВИТЬ { н-л / РАБОТУ }

Глаголы перемещения данных

ПОМЕСТИТЬ { н-д-1 / литерал / ф-к / СЧЕТЧИК } В { н-д-3 / СЧЕТЧИК }

[, { н-д-4 / СЧЕТЧИК } ...]

ПРОСМОТРЕТЬ н-д.

{ СЧИТАЯ { ДО ПЕРВОГО / ВСЕ / ВЕДУЩИЕ } н-л-1 [ЗАМЕНЯЯ НА н-л-2] }

{ ЗАМЕНЯЯ { ВСЕ / ВЕДУЩИЕ / ДО ПЕРВОГО / ПЕРВЫЙ } н-л-3 НА н-л-4 }

УСТАНОВИТЬ { н-и-1 / н-д-1 } НА { н-и-2 / н-д-2 / ч-л-2 }

Форматы раздела процедур

Глаголы ввода-вывода

ОТКРЫТЬ { ВХОДНОЙ / ВЫХОДНОЙ } н-м-1 [БЕЗ ПЕРЕМОТКИ] [, н-м-2 [БЕЗ ПЕРЕМОТКИ] ...] [, { ВХОДНОЙ / ВЫХОДНОЙ } н-м-3 [БЕЗ ПЕРЕМОТКИ] [, н-м-4 [БЕЗ ПЕРЕМОТКИ] ...] ...]

ЧИТАТЬ н-м в КОНЦЕ повелительный-оператор

ПИСАТЬ название-записи [{ СПЕРВА / ЗАТЕМ } ПРОДВИГАЯ целое { СТРОК / СТРОКИ / СТРОКУ }]

ЗАКРЫТЬ н-м-1 [С ОСВОБОЖДЕНИЕМ] [, н-м-2 С ОСВОБОЖДЕНИЕМ ...]

ПРИНЯТЬ н-д.

ВЫДАТЬ { ч-л-1 / н-д-1 / н-л-1 / ф-к-1 } [, { ч-л-2 / н-д-2 / н-л-2 / ф-к-2 } ...] [НА у-н-1]

СОРТИРОВАТЬ н-м-1 ПО { ВОЗРАСТАНИЮ / УБЫВАНИЮ } КЛЮЧА н-д-1 [, н-д-2 ...] [, ПО { ВОЗРАСТАНИЮ / УБЫВАНИЮ } КЛЮЧА н-д-3 [, н-д-4 ...]...] [ПОЛУЧАЯ н-м-2]

Глаголы связи между программами

ВЫЗВАТЬ название-программы [ИСПОЛЬЗУЯ н-л-1, н-д-1 [, н-д-2 ...] [, н-л-2, н-д-3 [, н-д-4 ...] ...]]

ВЫЙТИ ИЗ-ПРОГРАММЫ

ОСВОБОДИТЬ

ВОЙТИ В название-программы [({ н-д-1 / ч-л-1 / н-л-1 } [, { н-д-2 / ч-л-2 / н-л-2 } ...])]

ВОЙТИ В название-программы [ИСПОЛЬЗУЯ н-л-1, н-д-1 [, н-д-2 ...] [, н-л-2, н-д-3 [, н-д-4 ...] ...]]

4.7. ПРИМЕР ЗАДАЧИ НА КОБОЛЕ

В настоящем параграфе приведен упрощенный пример задачи, которая состоит из двух транслируемых отдельно, но взаимодействующих в работе программ. Программа ПКОМП

вызывает программу ПЧВЕД. Первая из указанных программ производит раскомпоновку массива расчетных листков по заработной плате в два массива: массив документов для платежной ведомости и массив документов для бухгалтерских отчетов (сводов). Программа ПЧВЕД производит печать платежной ведомости по заработной плате по документам массива, полученного предыдущей программой, с использованием справочного массива. Программы не имеют общих областей. Ниже приведены тексты программ ПКОМП и ПЧВЕД.

```
РАЗДЕЛ ИДЕНТИФИКАЦИИ.
ПРОГРАММА. ПЕРЕКОМПОНОВКА РАСЧЕТНЫХ ЛИСТКОВ (ИДЕНТИ-
ФИКАТОР ПКОМП).
ДАТА НАПИСАНИЯ. ФЕВРАЛЬ 1973.
ЗАМЕЧАНИЯ. ПРОГРАММА ВЫЗЫВАЕТ ПРОГРАММУ ПЧВЕД.
РАЗДЕЛ ОБОРУДОВАНИЯ.
   СЕКЦИЯ КОНФИГУРАЦИИ.
      ИСХОДНАЯ-МАШИНА. МИНСК-32.
      РАБОЧАЯ-МАШИНА. МИНСК-32.
   СЕКЦИЯ ВВОДА-ВЫВОДА.
      УПРАВЛЕНИЕ-МАССИВАМИ.
         ДЛЯ РЛИСТ ПРЕДНАЗНАЧИТЬ МЛ 1.
         ДЛЯ СВОДЫ ПРЕДНАЗНАЧИТЬ МЛ 2.
         ДЛЯ ПЛВЕД ПРЕДНАЗНАЧИТЬ МЛ 3.
РАЗДЕЛ ДАННЫХ.
   СЕКЦИЯ МАССИВОВ.
ОМ РЛИСТ; В БЛОКЕ 50 ЗАПИСЕЙ; МЕТКИ СТАНДАРТНЫ, ШИФР МАС-
СИВА 'РЛИСТ'.
 01 ЛИСТ.
 02 ЦЕХ Ш 9(3).
 02 ТАБНОМ Ш 9(6).
 02 КАТЕГОРИЯ Ш 99.
 02 СИСТЕМА-ОПЛАТ Ш 9.
 02 НАЧИСЛЕНИЯ ПОВТОРЯЕТСЯ 20 РАЗ, ИНДЕКСИРУЕТСЯ ИНД1.
 03 ВИД-ОПЛАТ Ш 99.
 03 ВИД-УД Ш 99.
 03СУММА-НАЧ Ш 999Т99.
 02 УДЕРЖАНИЯ ПОВТОРЯЕТСЯ 15 РАЗ, ИНДЕКСИРУЕТСЯ ИНД2.
 03 СУММА-УД Ш 999Т99.
ОМ СВОДЫ В БЛОКЕ 200 ЗАПИСЕЙ; МЕТКИ СТАНДАРТНЫ, ШИФР
МАССИВА 'СВОДЫ'.
 01 СВОДЫ.
 02 ЗАП Ш ХХ ЗНАЧ '01'.
 02 ЦЕХС1 Ш 999.
 02 ТНОМС1 Ш 9(6).
 02 КАТЕГС1 Ш 99.
 02 СУММАС1 Ш 999Т99.
 02 ВИДОПЛС1 Ш 99.
 01 СВОД2.
 02 ЗАП Ш ХХ ЗНАЧ '02'.
 02 ЦЕХС2 Ш 999.
 02 ТНОМС2 Ш 9(6).
 02 ВУДС2 Ш 99.
 02 СУМУДС2 Ш 999Т99.
ОМ ПЧВЕД В БЛОКЕ 200 ЗАПИСЕЙ; МЕТКИ СТАНДАРТНЫ, ШИФР
МАССИВА 'ПЧВЕД'.
01 ВЕДОМОСТЬ.
```

```
 02 ЦЕХВ Ш 999.
 02 ТНОМВ Ш 9(6).
 02 СУММАВ Ш 3999Т99.
СЕКЦИЯ РАБОЧЕЙ-ПАМЯТИ.
77 СУММАНАЧИСЛ Ш 999Т99.
77 СУММАУДЕРЖ Ш 999Т99.
РАЗДЕЛ ПРОЦЕДУР.
НАЧ. ОТКРЫТЬ ВХОДНОЙ РЛИСТ, ВЫХОДНОЙ СВОДЫ, ВЫХОДНОЙ
ПЧВЕД.
ЧТ. ЧИТАТЬ РЛИСТ В КОНЦЕ ПЕРЕЙТИ К ЗАКР.
      ПОМЕСТИТЬ 0 В СУММАНАЧИСЛ, СУММАУДЕРЖ.
ФОРМ-С1. ВЫПОЛНИТЬ С1 МЕНЯЯ ИНД1 ОТ 1 НА 1 ДО ИНД1=21.
      ПЕРЕЙТИ К ФОРМ-С2.
С1. ЕСЛИ СУММА-НАЧ(ИНД1)=0 ПЕРЕЙТИ К ФОРМ-С2.
      ПОМЕСТИТЬ ЦЕХ В ЦЕХС1, ПОМЕСТИТЬ ТАБНОМ В ТНОМС1.
      ПОМЕСТИТЬ КАТЕГОРИЯ В КАТЕГС1.
ПОМЕСТИТЬ СУММА-НАЧ (ИНД1) В СУММАС1.
ПОМЕСТИТЬ ВИД-ОПЛАТ(ИНД1) В ВИДОПЛС1.
      ПИСАТЬ СВОД1.
      ВЫЧИСЛИТЬ  СУММАНАЧИСЛ=СУММАНАЧИСЛ+СУММА-НАЧ-
(ИНД1).
ФОРМ-С2. ВЫПОЛНИТЬ С2 МЕНЯЯ ИНД2 ОТ 1 НА 1 ДО ИНД2=16.
      ПЕРЕЙТИ К ФОРМ-ВЕД.
С2. ЕСЛИ СУММА-УД(ИНД2)=0 ПЕРЕЙТИ К ФОРМ-ВЕД.
      ПОМЕСТИТЬ ЦЕХ В ЦЕХС2.
      ПОМЕСТИТЬ ТАБНОМ В ТНОМС2.
ПОМЕСТИТЬ ВИД-УД(ИНД2) В ВУДС2.
      ПОМЕСТИТЬ СУММА-УД(ИНД2) В СУМУДС2.
      ПИСАТЬ СВОД2.
      ВЫЧИСЛИТЬ  СУММАУДЕРЖ=СУММАУДЕРЖ+СУММА-УД
(ИНД2).
ФОРМ-ВЕД. ПОМЕСТИТЬ ЦЕХ В ЦЕХВ, ПОМЕСТИТЬ ТАБНОМ В
      ТНОМВ.
      ВЫЧИСЛИТЬ СУММАВ=СУММАНАЧИСЛ-СУММАУДЕРЖ.
      ПИСАТЬ ВЕДОМОСТЬ.
      ПЕРЕЙТИ К ЧТ.
ЗАКР. ЗАКРЫТЬ РЛИСТ, СВОДЫ, ПЛВЕД С ОСВОБОЖДЕНИЕМ.
      ВЫЗВАТЬ ПЧВЕД. ОСВОБОДИТЬ.
      ОСТАНОВИТЬ РАБОТУ.
РАЗДЕЛ ИДЕНТИФИКАЦИИ.
ПРОГРАММА. ПЕЧАТЬ ПЛАТЕЖНОЙ ВЕДОМОСТИ (ИДЕНТИФИКАТОР
ПЧВЕД).
АВТОР. НИКОЛАЕВ.
ДАТА НАПИСАНИЯ. ФЕВРАЛЬ 1973.
ЗАМЕЧАНИЯ. ПРОГРАММА ВЫЗЫВАЕТСЯ ПРОГРАММОЙ ПКОМП.
РАЗДЕЛ ОБОРУДОВАНИЯ.
    СЕКЦИЯ КОНФИГУРАЦИИ.
      ИСХОДНАЯ-МАШИНА. МИНСК-32.
      РАБОЧАЯ-МАШИНА. МИНСК-32.
    СЕКЦИЯ ВВОДА-ВЫВОДА.
      УПРАВЛЕНИЕ-МАССИВАМИ.
            ДЛЯ ПЛВЕД ПРЕДНАЗНАЧИТЬ МЛ 3.
            ДЛЯ СПРАВОЧНЫЙ ПРЕДНАЗНАЧИТЬ МЛ 4.
РАЗДЕЛ ДАННЫХ.
    СЕКЦИЯ МАССИВОВ.
ОМ ПЛВЕД; В БЛОКЕ 200 ЗАПИСЕЙ; МЕТКИ СТАНДАРТНЫ, ШИФР
МАССИВА 'ПЧВЕД'.
01 ВЕДОМ.
02 ЦЕХ Ш 999.
```

```
 02 ТНОМВ Ш 9(6).
 02 СУММАВ Ш 3999Т99.
ОМ СПРАВОЧНЫЙ; В БЛОКЕ 200 ЗАПИСЕЙ; МЕТКИ СТАНДАРТНЫ
ШИФР МАССИВА 'СПРАВ'.
01 СПРАВКА.
 02 ЦЕХ Ш 999.
 02 ТАБНОМ Ш 9(6).
 02 ФАМИЛИЯ Ш Х(20).
СЕКЦИЯ РАБОЧЕЙ-ПАМЯТИ.
77 ЧЕРТА Ш Х(80) ЗНАЧ ВСЕ '—'.
77 МЕСЯЦГОД Ш Х(13).
77 ИТОГ Ш 9(5)Т99.
77 ИТОГРЕД Ш 9(5) 'РУБ'. 99 'КОП'.
77 СЧЛИСТ Ш 99.
77 СЧСТРОК Ш 99.
77 ПРОБ38 Ш Х(38) ЗНАЧ ПРОБЕЛ.
77 СУМРЕД Ш 999 'РУБ'. 99 'КОП'.
77 ПРОБ29 Ш Х(29) ЗНАЧ ПРОБЕЛ.
77 ПРОБ20 Ш Х(20) ЗНАЧ ПРОБЕЛ.
РАЗДЕЛ ПРОЦЕДУР.
    ПРИНЯТЬ МЕСЯЦГОД. ПОМЕСТИТЬ 0 В ИТОГ.
    ПОМЕСТИТЬ 1 В СЧЛИСТ.
БЛО. ВЫДАТЬ ПРОБ38, '—', СЧЛИСТ, '—'.
    ПОМЕСТИТЬ 1 В СЧСТРОК.
ОТКР. ОТКРЫТЬ ВХОДНОЙ ПЛВЕД, ВХОДНОЙ СПРАВОЧНЫЙ.
    ВЫПОЛНИТЬ БЛ6 3 РАЗА.
БЛ2. ВЫДАТЬ ПРОБ29, 'ПЛАТЕЖНАЯ ВЕДОМОСТЬ'.
    ВЫДАТЬ ПРОБ38, МЕСЯЦГОД.
    ВЫПОЛНИТЬ БЛ6 2 РАЗА.
БЛ3. ВЫДАТЬ ЧЕРТА. ВЫДАТЬ ПРОБ20, ПРОБ20, '*'.
    ВЫДАТЬ '_____Ф.И.О.', ПРОБ20, '*', ПРОБ20,
'СУММА'.
    ВЫДАТЬ ПРОБ20, ПРОБ20, '*'.
    ВЫДАТЬ ЧЕРТА.
    ВЫЧИСЛИТЬ СЧСТРОК=СЧСТРОК+7.
ЧТВЕД. ЧИТАТЬ ПЛВЕД В КОНЦЕ ПЕРЕЙТИ К ИТОГО.
ЧТСП. ЧИТАТЬ СПРАВОЧНЫЙ В КОНЦЕ ПЕРЕЙТИ К ИТОГО.
    ЕСЛИ ЦЕХВ=ЦЕХ ПЕРЕЙТИ К БЛ1.
    ЕСЛИ ЦЕХВ>ЦЕХ ПЕРЕЙТИ К ЧТСП. ПЕРЕЙТИ К ЧТВЕД.
БЛ1. ЕСЛИ ТНОМВ=ТАБНОМ ПЕРЕЙТИ К БЛ4.
    ЕСЛИ ТНОМВ>ТАБНОМ ПЕРЕЙТИ К ЧТСП. ПЕРЕЙТИ К ЧТВЕД.
БЛ4. ПОМЕСТИТЬ СУММАВ В СУМРЕД.
    ВЫЧИСЛИТЬ ИТОГ=ИТОГ+СУММАВ.
    ВЫДАТЬ '_____', ФАМИЛИЯ, '_____*'.
    ПРОБ20, СУМРЕД.
    ВЫЧИСЛИТЬ СЧСТРОК=СЧСТРОК+1.
    ЕСЛИ СЧСТРОК=65 ПЕРЕЙТИ К БЛ5.
БЛ5. ВЫПОЛНИТЬ БЛ6 5 РАЗ ПЕРЕЙТИ К БЛ7.
    ПЕРЕЙТИ К ЧТВЕД.
БЛ6. ВЫДАТЬ ПРОБЕЛ.
БЛ7. ВЫЧИСЛИТЬ СЧЛИСТ=СЧЛИСТ+1, ВЫПОЛНИТЬ БЛ0.
    ПЕРЕЙТИ К БЛ3.
ИТОГО. ПОМЕСТИТЬ ИТОГ В ИТОГРЕД.
    ВЫДАТЬ ПРОБ38, 'И_Т_О_Г_О_*', ИТОГРЕД.
    ЗАКРЫТЬ ПЛВЕД, СПРАВОЧНЫЙ.
КОН. ВЫЙТИ ИЗ-ПРОГРАММЫ.
```

Глава 5

МЕТОДЫ РЕШЕНИЯ ЭКСТРЕМАЛЬНЫХ ЗАДАЧ

5.1. ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Линейное программирование изучает важную для практики задачу отыскания максимума (минимума) линейной функции при наличии ограничений в виде линейных неравенств или уравнений. В настоящем подразделе излагается симплекс — метод решения задачи линейного программирования [15].

1. *Модифицированные жордановы исключения*

Пусть рассматривается система

$$y_i = a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n,\ i = 1, 2, \ldots, m \qquad (5.1)$$

из m линейных форм с n независимыми переменными. Перепишем эту систему в виде

$$y_i = -a_{i1}(-x_1) - a_{i2}(-x_2) - \ldots - a_{in}(-x_n),\ i = 1, 2, \ldots, m. \qquad (5.1')$$

Эта система может быть записана в виде табл. 5.1. В этой таблице

$$\alpha_{ik} = -a_{ik},\ i = 1, 2, \ldots, m,\ k = 1, 2, \ldots, n.$$

Таблица 5.1

	$-x_1$	$-x_2$	$\ldots$	$-x_s$	$\ldots$	$-x_n$
y_1	α_{11}	α_{12}	$\ldots$	α_{1s}	$\ldots$	α_{1n}
$\ldots$	$\ldots$	$\ldots$		$\ldots$		$\ldots$
y_r	α_{r1}	α_{r2}	$\ldots$	$\boxed{\alpha_{rs}}$	$\ldots$	α_{rn}
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
y_m	α_{m1}	α_{m2}		α_{ms}		α_{mn}

Будем называть шагом модифицированного жорданова исключения, произведенным над табл. 5.1 с разрешающим элементом $\alpha_{rs} \neq 0$, *с* r-й разрешающей строкой и s-м разрешающим

столбцом, схематизированную операцию перемены ролями зависимой переменной y_r с независимой x_s. Эта операция сводится к решению уравнения

$$y_r = -a_{r1}(-x_1) - a_{r2}(-x_2) - \ldots - a_{rn}(-x_n)$$

относительно x_s, подстановке x_s во все остальные уравнения системы (5.1′) и записи полученной системы в виде новой таблицы 5.2, аналогичной табл. 5.1. В табл. 5.2

$$b_{ij} = \frac{a_{ij}a_{rs} - a_{is}a_{rj}}{a_{rs}} \quad (i \neq r,\ j \neq s).$$

Таблица 5.2

	$-x_1$	$-x_2$	$\ldots$	$-y_r$	$\ldots$	$-x_n$
y_1	b_{11}	b_{12}	$\ldots$	$-\frac{a_{1s}}{a_{rs}}$	$\ldots$	b_{1n}
$\ldots$	$\ldots$	$\ldots$		$\ldots$		$\ldots$
x_s	$\frac{a_{r1}}{a_{rs}}$	$\frac{a_{r2}}{a_{rs}}$	$\ldots$	$\frac{1}{a_{rs}}$	$\ldots$	$\frac{a_{rn}}{a_{rs}}$
$\ldots$	$\ldots$	$\ldots$		$\ldots$		$\ldots$
y_m	b_{m1}	b_{m2}	$\ldots$	$-\frac{a_{ms}}{a_{rs}}$	$\ldots$	b_{mn}

Таким образом, один шаг модифицированного жорданова исключения с разрешающим элементом a_{rs} переводит табл. 5.1 в новую табл. 5.2 по следующим правилам:

1) разрешающий элемент заменяется обратной величиной;

2) остальные элементы разрешающей строки делятся на разрешающий элемент;

3) остальные элементы разрешающего столбца делятся на разрешающий элемент и меняют знаки на противоположные;

4) элементы $b_{ij}(j \neq s,\ i \neq r)$ вычисляются по формуле

$$b_{ij} = a_{ij} - \frac{a_{is}a_{rj}}{a_{rs}}.$$

2. *Задача линейного программирования. Геометрическая интерпретация*

Задача линейного программирования состоит в максимизации линейной формы

$$z = p_1x_1 + p_2x_2 + \ldots + p_nx_n \tag{5.2}$$

при ограничениях

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leqslant a_i,\ i = 1, 2, \ldots, m, \tag{5.3}$$

$$x_j \geqslant 0,\ j = 1, 2, \ldots, n. \tag{5.4}$$

Систему линейных неравенств (5.3) можно переписать в виде

$$y_i = -a_{i1}x_1 - a_{i2}x_2 - \ldots - a_{in}x_n + a_i \geqslant 0, \qquad (5.5)$$
$$i = 1, 2, \ldots, m.$$

Тогда задача состоит в максимизации формы (5.2) при условиях (5.4), (5.5).

Задачу линейного программирования можно легко интерпретировать геометрически. Каждое неравенство

$$y_i = -a_{i1}x_1 - a_{i2}x_2 - \ldots - a_{in}x_n + a_i \geqslant 0$$

определяет в евклидовом n-мерном пространстве полупространство, состоящее из точек $x(x_1, \ldots, x_n)$, расположенных по одну сторону от плоскости

$$y_i = -a_{i1}x_1 - a_{i2}x_2 - \ldots - a_{in}x_n = 0$$

и на самой этой плоскости. Точки, принадлежащие всем полупространствам (5.5), образуют некоторый выпуклый многогранник Ω.

Значение функции

$$z(x) = p_1x_1 + p_2x_2 + \ldots + p_nx_n$$

в точке $x'(x'_1, \ldots, x'_n)$ можно рассматривать как уклонение точки $x'(x'_1, \ldots, x'_n)$ от плоскости

$$p_1x_1 + p_2x_2 + \ldots + p_nx_n = 0, \qquad (5.6)$$

понимая под уклонением данной точки от этой плоскости число, которое получим, подставив в левую часть уравнения (5.6) вместо $x_1, x_2, \ldots, x_n$ координаты $x'_1, x'_2, \ldots, x'_n$ этой точки.

Таким образом, геометрический смысл задачи линейного программирования заключается в отыскании в многограннике Ω точки, которая наиболее уклонена от плоскости (5.6).

В случае двумерного пространства имеем картину, изображенную на рис. 5.1.

Здесь многогранником Ω является многоугольник, плоскостями $y_i = -a_{i1}x_1 - a_{i2}x_2 + a_i = 0$ — прямые, полупространствами $y_i \geqslant 0$ — полуплоскости (на рис. 5.1 они отмечены штриховкой).

Ясно, что решением задачи линейного программирования будет какая-то вершина многогранника Ω. На рис. 5.1 решение задачи максимизации дает вершина P_1,

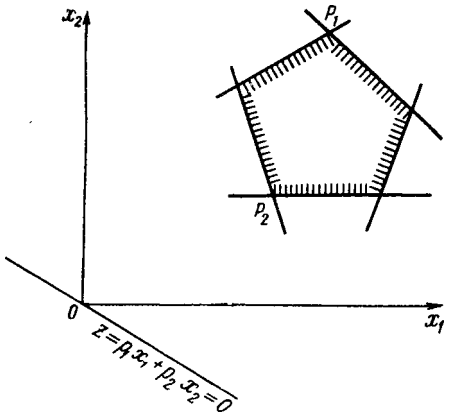


Рис. 5.1. Геометрическая интерпретация задачи линейного программирования на плоскости

а задачи минимизации — вершина P_2, причем эти решения единственны.

Возможны также случаи существования бесчисленного множества решений, случай неограниченности функции z на Ω и, наконец, случай отсутствия решения.

Симплекс-метод состоит из алгоритма отыскания какого-нибудь опорного решения среди решений системы линейных неравенств (5.3), т. е. решения — вершины многогранника Ω (или из установления факта несовместимости системы), и из алгоритма последовательного перехода от полученного уже опорного решения системы (5.3) к новому опорному решению, для которого линейная форма имеет большее значение до получения максимизирующего, т. е. оптимального, решения.

3. Отыскание опорного решения

Форму (5.2) и условия (5.5) запишем в виде табл. 5.3.

Таблица 5.3

	$-x_1$	$-x_2$	...	$-x_n$	1
y_1	a_{11}	a_{12}	...	a_{1n}	a_1
...	...	...		...	...
y_m	a_{m1}	a_{m2}	...	a_{mn}	a_m
$z=$	$-p_1$	$-p_2$	...	$-p_n$	0

Пусть $a_1 \geqslant 0$, $a_2 \geqslant 0$, ..., $a_m \geqslant 0$. В этом случае табл. 5.3 дает возможность сразу получить одно из опорных решений системы (5.5). Это будет решение (вершина многогранника Ω), определяемое равенствами

$$x_1 = 0,\ x_2 = 0,\ \ldots,\ x_n = 0,$$

так как тогда

$$y_1 = a_1 \geqslant 0,\ \ldots,\quad y_m = a_m \geqslant 0$$

и, следовательно, все неизвестные неотрицательны и удовлетворяется система (5.3).

Пусть в табл. 5.3 есть хотя бы один отрицательный свободный член, например, пусть $a_r < 0$. Теперь значения

$$x_1 = x_2 = \ldots = x_n = 0$$

не дают никакого решения системы (5.5), так как при этих значениях имеем $y_r = a_r < 0$.

Симплекс-метод для отыскания опорного решения означает специальное правило для перехода от данной вершины $x_1 = x_2 = \ldots = x_n = 0$ к такой соседней, которую отделяет от много-

гранника Ω меньшее число плоскостей, т. е. для которой в соответствующей таблице содержится меньшее число отрицательных свободных членов.

Для осуществления перехода от вершины $x_1=x_2=\ldots=x_n=0$ к указанной соседней производим шаг модифицированного жорданова исключения, выбирая разрешающий элемент согласно следующему правилу.

1) Выбираем строку с отрицательным свободным членом (пусть, например, $a_r<0$). Если среди коэффициентов этой строки нет отрицательных, то система (5.3) несовместна.

2) Если среди коэффициентов рассматриваемой строки есть отрицательные, то берем какой-нибудь из них (пусть $a_{rs}<0$) и столбец, содержащий этот коэффициент, берем в качестве разрешающего.

3) Выбор разрешающей строки производится так: вычисляем все неотрицательные отношения $\frac{a_i}{a_{is}} \geqslant 0$ свободных членов к соответствующим отличным от нуля коэффициентам разрешающего столбца, находим среди них наименьшее и элемент $a_{i_0 s}$, для которого оно достигается, берем в качестве разрешающего.

В случае вырождения, когда $\min \frac{a_i}{a_{is}} = \frac{a_{i_0}}{a_{i_0 s}} = 0$, выбираем $a_{i_0 s}$ в качестве разрешающего лишь при $a_{i_0 s} > 0$.

Если в результате применения предыдущего правила разрешающим оказался элемент a_{rs}, то после шага модифицированного жорданова исключения новый свободный член a_r рассматриваемой r-й строки станет уже положительным

$$a_r' = \frac{a_r}{a_{rs}} > 0.$$

Если же разрешающим оказался коэффициент a_{ls}, где $l \neq r$, то новый свободный член a_r r-й строки останется еще отрицательным, так что поставленная цель освобождения от отрицательного свободного члена пока не достигнута. В этом случае продолжаем работать с этой (r-й) строкой, применяя к ней предыдущее правило и производя шаги модифицированных жордановых исключений до тех пор, пока либо не установим несовместность системы (5.3) (все коэффициенты этой строки станут неотрицательными), либо не избавимся от отрицательности его свободного члена (разрешающий элемент окажется из этой строки).

Так поступаем последовательно со всеми строками, в которых свободные члены отрицательны. После конечного числа шагов либо установим несовместность системы (5.3), либо придем к таблице, не содержащей отрицательных свободных членов, т. е. получим опорное решение нашей системы, приравняв к нулю все неизвестные, оказавшиеся на верху таблицы.

4. Отыскание оптимального решения задачи линейного программирования

Пусть рассматривается задача максимизации линейной формы (5.2)

$$z=p_1x_1+p_2x_2+\ldots+p_nx_n$$

при ограничениях (5.3), (5.4):

$$y_i=-a_{i1}x_1-a_{i2}x_2-\ldots-a_{in}x_n+a_i\geqslant 0,$$
$$i=1, 2, \ldots, m,$$
$$x_j\geqslant 0,\ j=1, 2, \ldots, n,$$

и пусть после отыскания опорного решения получена табл. 5.4.

Таблица 5.4

	$-y_1$	$-y_2$	$\ldots$	$-y_s$	$\ldots$	$-y_n$	
$y_{n+1}=$	$b_{n+1,1}$	$b_{n+1,2}$	$\ldots$	$b_{n+1,s}$	$\ldots$	$b_{n+1,n}$	b_{n+1}
$\ldots$	$\ldots$	$\ldots$		$\ldots$		$\ldots$	$\ldots$
$y_r=$	b_{r1}	b_{r2}	$\ldots$	b_{rs}	$\ldots$	b_{rn}	b_r
$\ldots$	$\ldots$	$\ldots$		$\ldots$		$\ldots$	$\ldots$
$y_m=$	b_{m1}	b_{m2}	$\ldots$	b_{ms}	$\ldots$	b_{mn}	b_m
$z=$	q_1	q_2	$\ldots$	q_s	$\ldots$	q_n	Q

В зависимости от знаков коэффициентов z-строки необходимо различать следующие два возможных случая.

Если $q_1\geqslant 0$, $q_2\geqslant 0$, ..., $q_n\geqslant 0$, то задача линейного программирования решена, причем

$$\max z=Q$$

и достигается в точке

$$y_1=y_2=\ldots=y_n=0.$$

Действительно, в этой точке имеем

$$y_{n+1}=b_{n+1}\geqslant 0,\ \ldots,\ y_m=b_m\geqslant 0,$$

т. е. удовлетворяются ограничения (5.3), и так как в любой другой точке многогранника Ω все y_1, y_2, ..., y_n неотрицательны, то

$$z=-q_1y_1-\ldots-q_ny_n+Q\leqslant Q,$$

т. е. Q — максимальное значение z.

Пусть среди коэффициентов z-строки есть отрицательные, например, пусть $q_s<0$. В этом случае, очевидно, уже нельзя утверждать, что в точке $y_1=\ldots=y_n=0$ значение Q функции z является максимальным. Действительно, если, например, точка $y_1=\ldots=y_{s-1}=0$, $y_s>0$, $y_{s+1}=\ldots=y_n=0$ удовлетворяет ограничениям (5.3) (т. е. $y_i=-b_{is}y_s+b_i\geqslant 0$, $i=n+1, \ldots, m$), то в этой точке $z=-q_sy_s+Q>Q$.

Для отыскания оптимального решения симплекс-метод определяет специальное правило перехода от полученной точки $y_1 = \ldots = y_n = 0$ (вершины многогранника Ω) к той соседней вершине этого многогранника, в которой значение z больше (не меньше) Q. Этот процесс продолжается, пока не будет найдена вершина, в которой значение z максимально, т. е. для которой все коэффициенты z-строки будут неотрицательны (или пока не будет установлено, что функция z не ограничена сверху).

Чтобы осуществить переход от вершины $y_1 = y_2 = \ldots = y_n = 0$ к упомянутой соседней вершине, делаем один шаг модифицированного жорданова исключения со следующим правилом выбора разрешающего элемента.

1) В качестве разрешающего берем столбец, содержащий отрицательный элемент z-строки (в рассматриваемом случае s-й столбец).

2) Отбираем все положительные коэффициенты этого столбца (если такие имеются), делим на них соответствующие свободные члены, сравниваем полученные отношения, и в качестве разрешающего берем тот из коэффициентов, для которого отношение имеет наименьшее значение (если их окажется больше одного, то берем любой из них).

После шага модифицированного жорданова исключения с разрешающим элементом, выбранным по только что сформулированному правилу, знак у q_s изменится на противоположный, так что новый коэффициент q_s окажется уже положительным. Если все остальные новые коэффициенты z-строки неотрицательны, то задача решена. Если же среди остальных коэффициентов новой z-строки есть отрицательные, то поступаем с каждым из них так, как с q_s, и после конечного числа шагов придем либо к случаю, когда в z-строке не окажется отрицательных коэффициентов (задача решена), либо к случаю отсутствия положительных коэффициентов в некотором столбце, содержащем отрицательный коэффициент z-строки, что означает неограниченность сверху функции z.

Действительно, пусть, например, $q_s < 0$ и среди коэффициентов s-го столбца нет положительного. В этом случае можно положить $y_1 = \ldots = y_{s-1} = 0$, $y_s = t > 0$, $y_{s+1} = \ldots = y_n = 0$. Тогда при $i \geqslant n+1$ получим

$$y_i = -b_{is}t + b_i \geqslant 0,$$

т. е. при любом $t \geqslant 0$ удовлетворяется система (5.3). Соответствующее же значение функции

$$z = -q_s t + Q$$

можно сделать сколь угодно большим при достаточно большом t.

5. Разные способы задания ограничений

Часто задача линейного программирования встречается в следующей формулировке.

Среди решений системы

$$\left.\begin{aligned} &a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = a_i,\ i = 1,\ 2,\ \ldots,\ m, \\ &x_j \geqslant 0,\ j = 1,\ 2,\ \ldots,\ n \end{aligned}\right\} \qquad (5.7)$$

найти то, которое максимизирует линейную форму (5.2)

$$z = p_1x_1 + p_2x_2 + \ldots + p_nx_n.$$

Предположим, что все $a_i \geqslant 0$, ибо мы всегда можем этого добиться.

Составляем жорданову табл. 5.5.

Таблица 5.5

	$-x_1$	$-x_2$	$\ldots$	$-x_n$	1
0	a_{11}	a_{12}	$\ldots$	a_{1n}	a_1
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
0	a_{m1}	a_{m2}	$\ldots$	a_{mn}	a_m
$z=$	$-p_1$	$-p_2$	$\ldots$	$-p_n$	0

Чтобы получить опорное решение, надо избавиться от всех 0-строк. Правило избавления от i-й 0-строки формулируется так: просматриваем столбец полученной таблицы, содержащий положительный коэффициент a_{ij} из i-й 0-строки; отмечаем все положительные коэффициенты этого столбца; делим на отмеченные коэффициенты соответствующие свободные члены и тот из коэффициентов, для которого это отношение наименьшее, берем в качестве разрешающего. Если этим коэффициентом окажется a_{ij}, то после шага модифицированного жорданова исключения избавляемся от 0-строки.

Если же разрешающим оказался элемент a_{lj} $(l \neq i)$, то продолжаем шаги модифицированных жордановых исключений над i-й строкой до тех пор, пока либо избавимся от i-й 0-строки, либо установим несовместность системы (5.7) (все коэффициенты i-й 0-строки станут неположительными при положительном свободном члене).

После того как нуль переброшен наверх, столбец под ним вычеркиваем.

Изложенный способ освобождения от 0-строк применяем до тех пор, пока не исключим все 0-строки или не установим несовместность системы (5.7).

Оптимальное решение находится точно так же, как в п. 4.

Иногда ограничения имеют смешанный характер: они состоят из неравенств и ограничений. В этом случае задача линейно-

го программирования состоит в максимизации линейной формы

$$z=p_1x_1+p_2x_2+\ldots+p_nx_n$$

при ограничениях:

$$a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n\leqslant a_i,\ i=1,\ 2,\ \ldots,\ r;$$

$$a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n=a_i,\ i=r+1,\ \ldots,\ m;$$

$$x_j\geqslant 0,\ j=1,\ 2,\ \ldots,\ n.$$

Предполагается, что $a_i\geqslant 0$, $i=r+1,\ \ldots,\ m$.

В этом случае неравенства вводим в таблицу в виде

$$y_i=-a_{i1}x_1-a_{i2}x_2-\ldots-a_{in}x_n+a_i,\ i=1,\ 2,\ \ldots,\ r,$$

а уравнения — в виде 0-уравнений

$$0=-a_{i1}x_1-a_{i2}x_2-\ldots-a_{in}x_n+a_i,\ i=r+1,\ \ldots,\ m.$$

Получаем табл. 5.6.

Таблица 5.6

	$-x_1$	$-x_2$	$\ldots$	$-x_n$	1
y_1	a_{11}	a_{12}	$\ldots$	a_{1n}	a_1
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
y_r	a_{r1}	a_{r2}	$\ldots$	a_{rn}	a_r
0	$a_{r+1,1}$	$a_{r+1,2}$	$\ldots$	$a_{r+1,n}$	a_{r+1}
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
0	a_{m1}	a_{m2}	$\ldots$	a_{mn}	a_m
$z=$	$-p_1$	$-p_2$	$\ldots$	$-p_n$	0

Сначала наверх перебрасываются нули и нулевые столбцы вычеркиваются. Затем отыскивается опорное решение, как в п. 3, и оптимальное — как в п. 4.

Принципиальная блок-схема вычислительного процесса, реализующего симплекс-метод, приведена на рис. 5.2.

Пример. Найти максимальное значение линейной формы

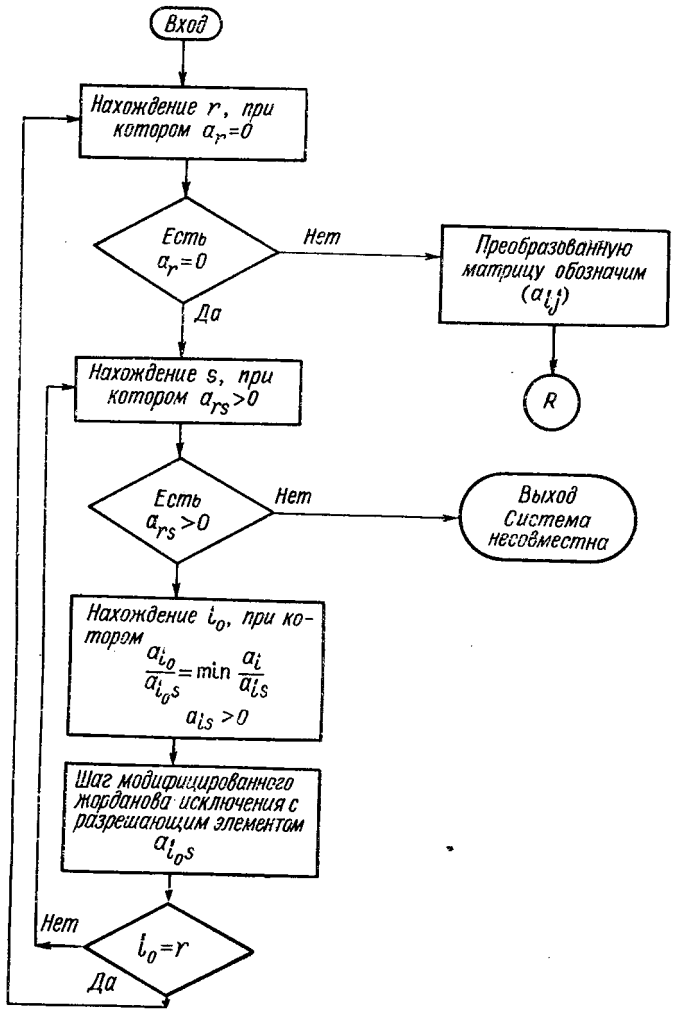
$$z=10x_1-x_2-9x_3-8x_4$$

при ограничениях

$$2x_1-x_2-3x_3-x_4+2=0,$$

$$5x_1-2x_2\qquad-3x_4+5=0,$$

$$-7x_1-4x_2-x_3-4x_4+1\geqslant 0,$$



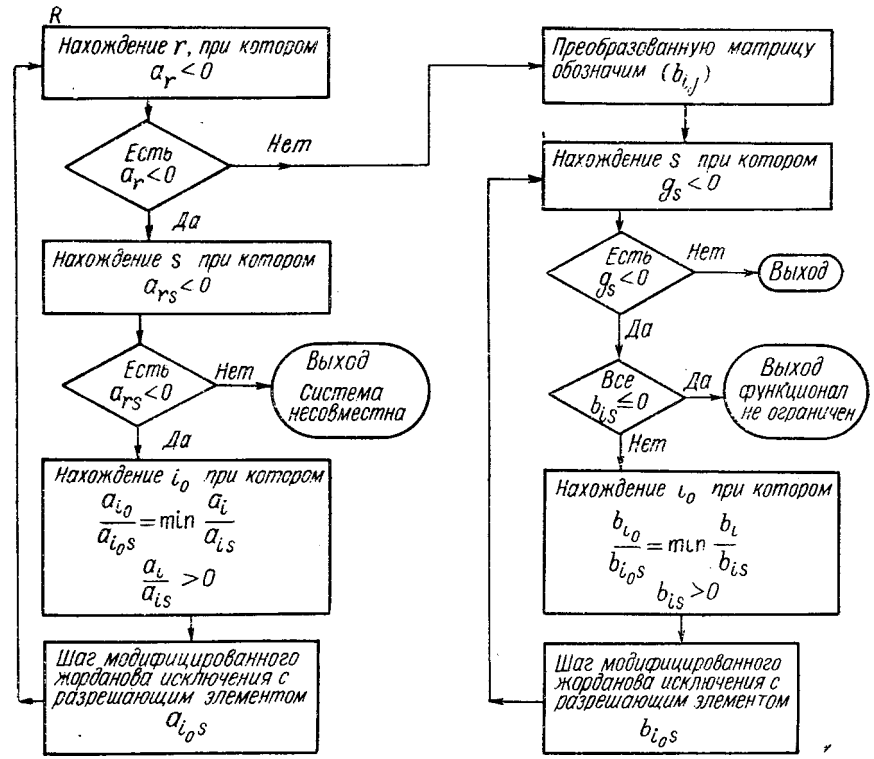


Рис. 5.2. Принципиальная блок-схема симплекс-метода

$$-3x_1-2x_2-5x_3-6x_4+10\geqslant 0,$$
$$x_1\geqslant 0,$$
$$x_2\geqslant 0,$$
$$x_3\geqslant 0,$$
$$x_4\geqslant 0.$$

Составляем табл. 1.

Таблица 1

	$-x_1$	$-x_2$	$-x_3$	$-x_4$	1
0	−2	[1]	3	1	2
0	−5	2	0	3	5
y_1	7	−4	1	4	1
y_2	3	2	5	6	10
z	−10	1	9	8	0

Сделав шаг модифицированного жорданова исключения с разрешающим элементом 1 из первой строки (взятым в рамку) и вычеркнув столбец коэффициентов под перенесенным наверх нулем, получим табл. 2.

Таблица 2

	$-x_1$	$-x_3$	$-x_4$	
x_2	−2	3	1	2
0	−1	−6	[1]	1
y_1	−1	13	8	9
y_2	7	−1	4	6
z	−8	6	7	−2

Аналогично освобождаемся от второй 0-строки. Приходим к табл. 3.

Теперь переходим к отысканию оптимального решения. Над отрицательным элементом z-строки разрешающим элементом будет 7, и после шага модифицированного жорданова исключения получаем табл. 4.

Здесь мы всю таблицу не вычисляли, так как все элементы z-строки неотрицательны. Следовательно, максимальное значение функционала задачи найдено:

$$\max z = -\frac{62}{7}.$$

Оно достигается при $x_3=0$, $y_1=0$.

Окончательное оптимальное решение будет следующим:

$$x_1 = \frac{1}{7},\quad x_2 = \frac{8}{7},\quad x_3 = 0,\quad x_4 = \frac{8}{7}.$$

Таблица 3

	$-x_1$	$-x_3$	1
x_2	—1	9	1
x_4	—1	—6	1
y_1	7	61	1
y_2	11	23	2
z	—1	48	—9

Таблица 4

	$-y_1$	$-x_3$	1
x_2			$^8/_7$
x_4			$^8/_7$
x_1			$^1/_7$
y_2			$^3/_7$
z	$\frac{1}{7}$	$\frac{397}{7}$	$\frac{-62}{7}$

5.2. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

1. Задача распределения ресурсов

Круг основных идей динамического программирования покажем на примере процесса распределения ресурсов и связанной с ним задачи оптимизации [5].

Допустим, что имеется некоторое количество экономических ресурсов. Под этим абстрактным термином могут пониматься люди, деньги, машины и т. д. Ресурсы можно употребить многими различными способами. Каждый такой возможный способ назовем технологическим процессом или производственным способом.

В результате употребления всех ресурсов или их части в каком-либо отдельном технологическом процессе будет получен некоторый доход. Размер дохода зависит как от употребленного количества ресурсов, так и от выбранного процесса.

Сделаем следующие предположения:

а) доходы, полученные от различных процессов, могут быть измерены общей единицей.

б) доход, полученный от любого данного процесса, не зависит от того, какие количества ресурсов были выделены для других процессов;

в) общий доход может быть определен как сумма доходов, полученных от отдельных процессов.

Задача состоит в распределении ресурсов таким образом, чтобы общий доход был максимальным.

Дадим теперь математическую формулировку описанной задачи. Пусть N различных процессов занумерованы в определенном порядке числами 1, 2, ..., N. Порядок нумерации процессов значения не имеет, но, будучи однажды принятым, он должен твердо сохраняться впредь. При этом если рассматриваются два процесса, то имеются в виду процессы 1 и 2; если рассматривается пять процессов, то имеются в виду процессы 1, 2, 3, 4, 5 и т. д.

Каждому процессу соответствует функция, выражающая зависимость дохода при этом процессе от количества выделенных для него ресурсов. Если x_i обозначает количество ресурсов, выделенных для i-го процесса, то через $g_i(x_i)$ обозначим соответствующий доход.

Предположения, касающиеся независимости процессов и аддитивности доходов, приводят к выражению

$$R(x_1,x_2,\ ...,\ x_N)=g_1(x_1)+g_2(x_3)+...+g_N(x_N) \tag{5.8}$$

для общего дохода от процесса распределения.

Задача максимизации возникает оттого, что в наличии обычно имеется лишь ограниченное количество ресурсов. Обозначая это количество через x, мы приходим к условию

$$x_1+x_2+...+x_N=x, \tag{5.9}$$

где $x_i\geqslant 0$. Следовательно, нужно максимизировать $R(x_1,x_2,...,x_N)$ при x_i, удовлетворяющих этим ограничениям.

Для того чтобы исследовать эту задачу, погружаем ее в некоторое семейство процессов распределения. Вместо рассмотрения одной задачи с данным количеством ресурсов и фиксированным числом процессов рассмотрим некоторое семейство таких задач, в которых x может принимать любые положительные значения и N — любые целые значения.

Поставим дополнительное требование, заключающееся в необходимости производить распределение ресурсов в каждую единицу времени. Сначала какое-то количество ресурсов назначается N-му процессу, затем $(N-1)$-му и т. д. Таким образом, вводится динамический процесс распределения. В результате статический, на первый взгляд, процесс распределения будет искусственно развернут во времени.

Перейдем к аналитическому рассмотрению этой задачи. Так как максимум $R(x_1,x_2,...,x_N)$ в указанной области зависит от x и N, сделаем эту зависимость явной, задав последовательность

функций $\{f_N(x)\}$, определенных для $N=1, 2,\ldots; x\geqslant 0$, следующим образом:

$$f_N(x) = \max_{\{x_i\}} R(x_1, \ldots, x_N),$$

где $x_i \geqslant 0$ и $\sum_{i=1}^{N} x_i = x$.

Функция $f_N(x)$ выражает оптимальный доход, получаемый от распределения количества ресурсов x по N процессам.

В двух частных случаях элементы последовательности $\{f_N(x)\}$ принимают особенно простой вид. Очевидно,

$$f_N(0) = 0, \quad N = 1, 2, \ldots,$$

если $g_i(0) = 0$ для любого i, что является разумным предположением. Также, очевидно,

$$f_1(x) = g_1(x)$$

для $x \geqslant 0$.

Легко найти рекуррентное соотношение, связывающее $f_N(x)$ и $f_{N-1}(x)$ для произвольных x и N. Пусть x_N, $0 \leqslant x_N \leqslant x_1$ — количество ресурсов, назначенное для N-го процесса. Тогда, каково бы ни было точное значение x_N, остающееся количество ресурсов $x - x_N$ будет использовано так, чтобы получить максимальный доход от остающихся $N-1$ процессов.

Так как этот оптимальный доход от распределения количества ресурсов $x - x_N$ по $N-1$ процессам по определению есть $f_{N-1}(x-x_N)$, назначение x_N для N-го процесса приводит к общему доходу

$$g_N(x_N) + f_{N-1}(x - x_N)$$

для модели с N процессами.

Очевидно, что оптимальным будет такой выбор x_N, который максимизирует эту функцию. Таким образом, получаем основное функциональное уравнение:

$$f_N(x) = \max_{0 \leqslant x_N \leqslant x} [g_N(x_N) + f_{N-1}(x - x_N)] \tag{5.10}$$

для $N = 2, 3, \ldots, x \geqslant 0$, причем

$$f_1(x) = g_1(x).$$

Для получения этих выводов был применен очень общий метод, известный под названием принципа оптимальности.

Принцип оптимальности. Оптимальное поведение обладает тем свойством, что, каковы бы ни были первоначальное состояние и решение в начальный момент, последующие решения должны составлять оптимальное поведение относительно состояния, получающегося в результате первого решения.

Для тех, кто не склонен доверять принципу оптимальности, дадим следующий вывод формулы (5.10). Замечая, что

$$\max_{\substack{x_1+x_2+\ldots+x_N=x \\ x_i\geqslant 0}} = \max_{0\leqslant x_N\leqslant x} \left[\max_{\substack{x_1+x_2+\ldots+x_{N-1}=x-x_N \\ x_i\geqslant 0}}\right],$$

получим цепочку равенств

$$f_N(x) = \max_{\substack{x_1+x_2+\ldots+x_N=x \\ x_i\geqslant 0}} [g_N(x_N) + g_{N-1}(x_{N-1}) + \ldots + g_1(x_1)] =$$

$$= \max_{0\leqslant x_N\leqslant x} \left[\max_{x_1+x_2+\ldots+x_{N-1}=x-x_N} [g_N(x_N) + g_{N-1}(x_{N-1}) + \ldots + g_1(x_1)]\right] =$$

$$= \max_{0\leqslant x_N\leqslant x} \left\{g_N(x_N) + \max_{\substack{x_1+x_2+\ldots+x_{N-1}=x-x_N \\ x_i\geqslant 0}} [g_{N-1}(x_{N-1}) + \ldots + g_1(x_1)]\right\} =$$

$$= \max_{0\leqslant x_N\leqslant x} [g_N(x_N) + f_{N-1}(x - x_N)],$$

что и дает требуемый результат.

Рекуррентное соотношение (5.10) дает теоретический метод для получения последовательности $\{f_N(x)\}$, если $f_1(x)$ известна. Действительно, $f_1(x)$ определяет $f_2(x)$ и т. д. Рассмотрим теперь вычислительную сторону использования рекуррентного соотношения (5.10). Чтобы задать в интервале $[0, x_0]$ все множество значений $f_N(x)$, воспользуемся их значениями, принимаемыми в конечном числе точек решетки

$$x=0,\ \Delta,\ 2\Delta,\ \ldots,\ R\Delta=x_0. \tag{5.11}$$

Каждый элемент последовательности $\{f_N(x)\}$ вычисляется в каждой из этих точек и только в этих точках. Значения $f_N(x)$ для x, отличных от точек решетки, будут получаться интерполяцией.

В задаче распределения, сформулированной выше, вполне разумно разрешить переменной x_N изменяться на том же самом множестве точек, на котором изменяется x. Поэтому в процессе максимизации x_N может принимать только значения, указанные в (5.11).

Когда $N=1$, функция $f_1(x)$ немедленно определяется равенством

$$f_1(x) = g_1(x).$$

Множество значений $\{f_1(k\Delta)\}$, $k=0,\ 1,\ \ldots,\ R$, с этого момента хранится в памяти вычислительной машины, что обеспечивает возможность вычислить $f_2(x)$ с помощью (5.10) для $N=2$, а именно:

$$f_2(x) = \max_{0\leqslant x_2\leqslant x} [g_2(x_2) + f_1(x - x_2)], \tag{5.12}$$

где x принимает только значения 0, Δ, 2Δ, ..., $R\Delta$. Так как никакой процесс перебора не может осуществить максимизацию на непрерывной области значений, необходимо заменить отрезок $[0, x]$ дискретным множеством. Тогда соотношение (5.12) заменится аппроксимирующим соотношением

$$f_2(x) = \max_{[k=0,\ 1,\ \ldots,R]} [g_2(k\Delta) + f_1(x - k\Delta)].$$

Процесс максимизации начинается с того, что вычисляются величины $g_2(0) + f_1(x)$ и $g_2(\Delta) + f_1(x-\Delta)$, затем эти величины сравниваются и большая из них запоминается. Далее вычисляется $g_2(2\Delta) + f_1(x-2\Delta)$ и сравнивается с большей из этих величин. Новую большую величину снова запоминаем и т. д. Процесс продолжается до тех пор, пока k не примет все допустимые значения. В результате получим $f_2(x)$ для данного значения x. В ходе этого процесса определяются не только значения $f_2(x)$ при $x=0$, Δ, ..., $R\Delta$, но также значение x_2, при котором в (5.12) достигается максимум. Обозначим его $x_2(x)$. В процессе вычислений запоминаются как $x_2(x)$, так и $f_2(x)$ для каждого x.

В итоге после двух шагов этого процесса можно заполнить таблицу типа табл. 5.7.

Таблица 5.7

x	$f_1(x)$	$x_1(x)$	$f_2(x)$	$x_2(x)$
0	—	—	—	—
Δ	—	—	—	—
2Δ	—	—	—	—
...	...	...	...	...
$R\Delta$	—	—	—	—

В этом случае $f_1(x) = g_1(x)$ и $x_1(x) = x$.

Приведенная табл. 5.7 дает решение двухшаговой задачи максимизации в следующем смысле. Если задано частное значение x, просматриваем столбец значений $x_2(x)$ до тех пор, пока не встретим соответствующее значение x_2. Как только это значение найдено, приходим к задаче определения оптимального распределения в одношаговом процессе с ресурсами $x - x_2(x)$.

Продолжая процесс N шагов, можно получить решение или в виде аналогичной таблицы, или даже в виде наборов x_N, x_{N-1}, ..., x_1, связанных с каждым значением N.

2. Задача размещения

Была рассмотрена задача оптимизации для случая, когда неизвестные x_i принимают значения из интервала $[0, x]$. Перейдем к случаю, когда неизвестные могут принимать значения из конечных множеств.

Рассмотрим задачу размещения в вариантной постановке, состоящую в следующем.

Необходимо определить для каждого i-го пункта ($i = 1, 2, \ldots, m$ такой объем производства x_i из заданного набора $A_i = (a_i^1, a_i^2, \ldots, a_i^{s_i})$, чтобы суммарные производственные затраты

$$P(x_1, x_2, \ldots, x_m) = \sum_{i=1}^{m} g_i(x_i) \quad (5.13)$$

были минимальными, а суммарный объем производства равен заданной величине c.

Таким образом, требуется найти минимум функционала (5.13) при условиях

$$x_1 + x_2 + \ldots + x_m = c, \quad (5.14)$$

$$x_i \in A_i, \; i = 1, 2, \ldots, m. \quad (5.15)$$

Покажем, как эту задачу можно решить методом динамического программирования. Как было показано раньше, задача (5.13)—(5.14) при условии $x_i \geqslant 0$, $i = 1, 2, \ldots, m$ решается при помощи функционального уравнения (5.10), причем в качестве точек решетки берутся числа $0, \Delta, 2\Delta, \ldots, R\Delta = c$. В случае, когда $x_i \in A_i$, $i = 1, 2, \ldots, m$, функциональное уравнение можно записать в виде

$$f_k(c) = \min_{x_k \in A_k} [g_k(x_k) + f_{k-1}(c - x_k)],$$

$$k = 2, 3, \ldots, \quad (5.16)$$

причем $f_1(c) = g_1(c)$.

При решении дискретных задач с помощью уравнения (5.16) существенное значение приобретает выбор точек решетки. Ниже приводятся условия, которым должны удовлетворять точки решетки.

Определение. О числе d будем говорить, что оно нормально относительно c и множества наборов $A_1, A_2, \ldots, A_m$, если существуют такие числа $a_i^{j_i} \in A_i$, $i = 1, 2, \ldots, m$ и число k, что

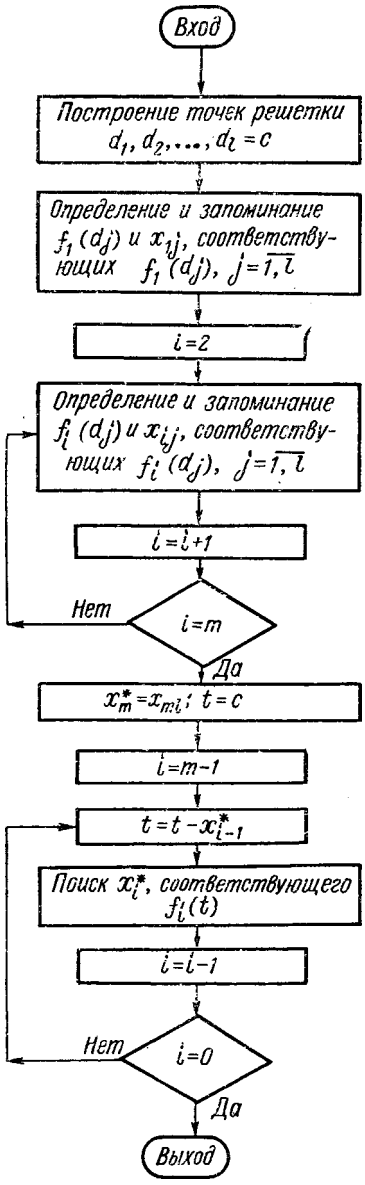


Рис. 5.3. Принципиальная блок-схема решения задач методом динамического программирования

$$\sum_{i=1}^{k} a_i^{\lambda} i = d, \quad d + \sum_{i=k+1}^{m} a_i^{\lambda} i = c.$$

Существует связь между нормальными числами и точками решетки, которая может быть сформирована в виде следующей теоремы.

Теорема 5.1. Если в качестве точек решетки взять все нормальные числа, то использование функционального уравнения (5.16) позволяет найти решение задачи (5.13) — (5.15).

Принципиальная блок-схема решения задач на ЭВМ приведена на рис. 5.3.

Проиллюстрируем работу алгоритма на примере.

Пример. Заданы три пункта размещения ($i=1,2,3$) по производству однородного продукта. Возможные мощности предприятий и затраты, связанные с размещением соответствующих мощностей, заданы в виде табл. 1. Суммарная потребность в продукте равна 5.

Таблица 1

$g_i(x_i)$ \ x	0	1	2	3
$g_1(x_1)$	0	2	3	5
$g_2(x_2)$	0	1	2	5
$g_3(x_3)$	0	2	3	4

В качестве точек решетки для решения данной задачи необходимо взять числа 0, 1, 2, 3, 4, 5.

Так как $f_1(x)=g_1(x)$, то

x	0	1	2	3	4	5
$f_1(x)$	0	2	3	5	—	—

Для нахождения $f_2(x)$ строим табл. 2.

Таблица 2

x_2 \ x	0	1	2	3	4	5
0	0	2	3	5	—	—
1	—	1	3	4	6	—
2	—	—	2	4	5	7
3	—	—	—	5	7	8

В строках табл. 2 расположены значения $f_2(x)$ при условиях, что $x_2=0$ (первая строка), $x_2=1$ (вторая строка) и т. д.

Из табл. 2 можно найти $f_2(x)$:

x	0	1	2	3	4	5
$f_2(x)$	0	1 1	2 2	4 1,2	6 1	7 2

Здесь указаны величины $f_2(x)$, а также значения x_2, при которых они достигаются.

Для нахождения $f_3(x)$ строим табл. 3.

Таблица 3

x / x_3	0	1	2	3	4	5
0	0	1	2	4	6	7
1	—	2	3	4	6	8
2	—	—	3	4	5	7
3	—	—	—	4	5	6

На основании табл. 3 находим $f_3(x)$:

x	0	1	2	3	4	5
$f_3(x)$	0	1 0	2 0	3 0,1 2,3	5 2,3	6 3

Следовательно, решение задачи достигается при $x_3=3$, $x_2=2$, $x_1=0$, причем $f_3(5)=6$.

5.3. КАЛЕНДАРНОЕ ПЛАНИРОВАНИЕ

1. Математическая постановка задач календарного планирования

Пусть на рассматриваемом производственном участке обработке подлежат n деталей d_i $(i=1, 2, ..., n)$. Обозначим некоторую произвольную операцию, которую необходимо выполнить

над d_i, через O_{ij} ($j=1,2, \ldots, m_i$). Каждая операция O_{ij} однозначно определяется двумя характеристиками

$$O_{ij} = <l_{ij},\ T_{ij}>,$$

где l_{ij} — номер группы оборудования, на котором может быть выполнена операция O_{ij};

T_{ij} — продолжительность выполнения операции на некотором эталонном для данной группы оборудования рабочем месте.

Задан технологический маршрут детали d_i, т. е. последовательность выполняемых операций

$$M_i = < O_{i1},\ O_{i2},\ \ldots,\ O_{im_i} >. \tag{5.17}$$

Предполагается, что эта последовательность выполняемых операций строго упорядочена. Операция O_{ij} должна выполняться без перерыва. Иными словами, если обозначить через $\underline{t}_{ij}$ время начала выполнения операции O_{ij}, а через $\overline{t}_{ij}$ — момент окончания выполнения операции, то для эталонного станка должно быть

$$\overline{t}_{ij} = \underline{t}_{ij} + T_{ij}.$$

В частности для технологического маршрута, заданного в виде (5.17), всегда должно быть

$$\overline{t}_{ij} \leqslant \underline{t}_{ij+1}, \tag{5.18}$$

т. е. всякая операция может выполняться не раньше окончания предыдущей операции.

Пусть весь рассматриваемый участок состоит из K рабочих мест R_k ($k=1, 2, \ldots, K$). Эти рабочие места объединяются в некоторые группы оборудования Γ_e ($e=1, 2, \ldots, L$), причем некоторые группы оборудования могут состоять только из одного станка и один и тот же станок может входить в несколько групп оборудования. Каждое рабочее место R_k характеризуется некоторым коэффициентом переработки норм по отношению к эталонному рабочему месту для данной группы. Поэтому реальное время выполнения операции O_{ij} на рабочем месте R_k из группы оборудования с номером l_{ij} будет равно $T_{ij} \cdot r_k(l_{ij})$ и, следовательно,

$$\overline{t}_{ij} = \underline{t}_{ij} + T_{ij}(k), \tag{5.19}$$

где $T_{ij}(k) = T_{ij} \cdot r_k(l_{ij})$.

Для упрощения будем считить, что ни на каком рабочем месте не может выполняться более одной операции одновременно. Это условие можно сформулировать и иначе: ни для каких двух операций $O_{i_1j_1}$ и $O_{i_2j_2}$, выполняемых на одном и том же рабочем месте, не может выполняться неравенство

$$\underline{t}_{i_1j_1} \leqslant t_{i_2j^2} < \overline{t}_{i_1j_1}. \tag{5.20}$$

Задача календарного планирования заключается в том, чтобы для описанного производственного участка построить календарный план, т. е. найти числа $\underline{t}_{ij}$, удовлетворяющие сформулированным условиям (5.18) — (5.20).

Совокупность чисел $\{t_i\}$ $(i=1, 2, ..., n;\ j=1, 2, ..., m_i)$, удовлетворяющую сформулированным ограничениям, будем называть календарным планом работы производственного участка или его графиком и обозначать символом G. Графиком $G(t_i)$ обработки детали d_i будем называть совокупность чисел $\{\underline{t}_{ij}\}$ $(j=1, 2, ..., m_i)$.

Очевидно, что существует бесчисленное множество графиков, удовлетворяющих сформулированным условиям. Например, если $G=\{\underline{t}_{ij}\}$ — график, то и $G=\{\underline{t}_{ij}+h\}$ тоже график $(h>0$ произвольно).

Таким образом, имеет смысл постановка вопроса о построении некоторого наилучшего графика в соответствии с выбранной функцией-критерием $F(G)$, т. е. графика, удовлетворяющего всем сформулированным условиям и доставляющего экстремальное (для определенности пусть минимальное) значение функции $F(G)$

$$F(\overline{G}) = \min_{G} F(G).$$

Укажем некоторые функции-критерии. Введем следующие определения.

Назовем временем готовности детали d_i величину v_i, если по условиям задачи

$$v_i \leqslant \underline{t}_{i1}.$$

Временем готовности рабочего места R_k назовем величину $B_k(0)$, если по условиям задачи

$$B_k(0) \leqslant \underline{t}_{ij}$$

для всех O_{ij} с привязкой к рабочему месту R_k. Величину D_i назовем требуемым сроком завершения обработки детали (в отличие от реального момента $\overline{t}_{im_i}$), если желательно, чтобы

$$\overline{t}_{im_i} \leqslant D_i.$$

Величину $z_i = \max\{0,\ t_{im_i} - D_i\}$ назовем общей задержкой детали d_i; $\Delta_{ij} = \underline{t}_{ij} - \underline{t}_{ij-1}$ — пролеживанием операции O_{ij}; $\Delta_i = \sum_{j=1}^{m_i} \Delta_{ij}$, где $\Delta_{i1} = \underline{t}_{i1} - v_i$, назовем общим пролеживанием детали d_i.

Назовем загрузкой рабочего места R_k в период времени, начиная с момента a до момента b, величину

$$T_k(a,\ b) = \sum_{i,\ j} T_{ij}(a,\ b),$$

где $\overline{T}_{ij}(a, b) = \mu\{[\underline{t}_{ij}, \overline{t}_{ij}] \cap [a, b]\}$;
$\mu\{[c, d]\}$ — длина отрезка $[c, d]$;

$A \cap B$ — общая часть отрезков A и B.

В введенных обозначениях могут быть записаны следующие функции-критерии:

$z = \sum_i z_i$ (задача минимизации суммарной задержки деталей);

$\overline{z} = \max_i \Delta_i$ (задача минимизации максимальной задержки деталей);

$\Delta = \sum_i \Delta_i$ (задача минимизации суммарного пролеживания деталей);

$T = \max_i \overline{t}_{im_i}$ (задача минимизации общего времени завершения обработки всех деталей);

$T(a, b) = \sum_k T_k(a, b)$ (задача максимизации общей загрузки участка в период времени, начиная с момента a до момента b).

Возможны и другие критерии оптимальности, в частности, связанные с минимизацией общей стоимости обработки деталей, минимизацией общего связывания денежных средств на участке и т. п. [41].

2. Методы решения задач календарного планирования

Формулировка общих задач календарного планирования показывает, что эти задачи относятся к задачам со сложными дискретными процессами оптимизации.

Точные методы, хотя бы принципиально решающие общие задачи календарного планирования, в настоящее время могут принести мало практической пользы в производственном управлении: настолько велики объемы вычислений при решении этими методами реальных задач. Только в самых простых случаях удается с уверенностью получить точное решение.

Наряду с разработкой точных методов совершенствуются различные методы и подходы приближенного решения задач календарного планирования. Это направление в настоящее время является наиболее продуктивным.

Кроме того, в решении задач календарного планирования оказываются эффективными методы моделирования, в том числе основанные на применении схем статистических испытаний.

В настоящее время нельзя остановиться на каком-то одном классе методов решения задач календарного планирования. Для одних задач исключительно эффективны методы динамического программирования или их дальнейшее развитие — методы последовательного конструирования, анализа и отбора вариантов, другие задачи могут решаться методами моделирования, некоторые классы задач могут быть успешно решены методами линейного программирования.

К схемам линейного программирования сводятся многие задачи, имеющие непосредственное отношение к оперативному пла-

нированию, например, такие, как задачи загрузки оборудования, задачи распределения заказов и др. Успехи линейного программирования вызвали многочисленные попытки представления задач календарного планирования в виде задач линейного программирования. Как правило, эти попытки приводят к задачам целочисленного линейного программирования больших размеров.

Задачи целочисленного линейного программирования весьма сложны для решения, и естественно, что на этом пути не получены сколько-нибудь заслуживающие внимания методы решения задач календарного планирования, хотя решение отдельных примеров и было продемонстрировано.

Попытки применения к решению задач календарного планирования методов динамического программирования привели к разработке исключительно эффективных схем решения для ряда простейших задач.

Еще более эффективными оказались методы последовательного конструирования, анализа и отбора вариантов, которые близки к методам динамического программирования, но не требуют выполнения принципа оптимальности. Если при последовательном конструировании вариантов удается ввести понятие доминирования одних вариантов над другими на основании сравнения отдельных частей вариантов, то можно построить простую схему нахождения оптимального решения.

Наиболее универсальным средством решения задач календарного планирования является моделирование. Применение методов моделирования основано на использовании при построении графиков правил предпочтения, а также рандомизированных правил предпочтения.

Актуальными являются попытки моделирования процесса оперативного планирования, который имеет место на производственных участках в реальных условиях. На производстве, как правило, определение сменного задания или корректировка его происходит на основании некоторых оценок предпочтительности одних операций перед другими. Это приводит к построению схем приближенного решения с помощью простых правил, основанных на применении функций предпочтения. Согласно этим правилам из всего множества готовых к обработке на каждой группе оборудования деталей запускаются в производство те, операции над которыми наиболее предпочтительны.

Подходы к решению задач календарного планирования при помощи функций предпочтения отличаются в основном логикой выбора каждой следующей операции из множества готовых к выполнению на данной группе оборудования. Несмотря на то, что в различных подходах этот выбор оказывается существенно различным, во всех случаях можно выделить правило

$$\pi(l, i_1, i_2, j_1, j_2),$$

согласно которому может быть установлено, какой из двух готовых к выполнению операций $O_{i_1 j_1}$ и $O_{i_2 j_2}$ деталей d_{i_1} и d_{i_2} на данной группе оборудования l должно быть отдано предпочтение. С точки зрения вычислений это предполагает возможность постановки в соответствие каждой готовой к выполнению операции O_{ij} величины $\omega(i, j, l)$ (функция предпочтения) и выбора из очереди операции с экстремальным значением функции предпочтения. Как правило, полагают

$$\omega(i, j, l) = \beta_i,$$

где β_i — некоторый «вес» детали, определяемый нормативными сроками изготовления партии, стоимостью обработки детали, трудоемкостью, и т. д.

Предлагалось также использовать в качестве функции предпочтения разность между временем выполнения первой и последней операции, соотношение времени очередной и следующей за нею операции, функции предпочтения, зависящие от простоя и пролеживания и др.

Общая схема решения задачи календарного планирования с помощью функций предпочтения выглядит следующим образом.

Во-первых, определенным образом обеспечивается фиксация всех «следующих» операций для построенного варианта решения, заключающегося в выполнении некоторых первых операций. Во-вторых, фиксируется некоторым образом общая картина загрузки оборудования. В-третьих, устанавливается некоторый способ выбора из общей очереди «следующих» операций с помощью функций предпочтения очередной операции, запускаемой в обработку. И, наконец, в-четвертых, формируется уже само решение задачи календарного планирования.

Изложенная общая схема предоставляет широкие возможности для учета конкретных особенностей производственного участка и условий машинной реализации.

3. Простейшие задачи календарного планирования. Задача двух станков

Задача двух станков формулируется следующим образом: требуется обработать за минимальное время n деталей d_i с технологическими маршрутами, состоящими только из двух операций продолжительностью a_i и b_i, при этом первая операция каждой детали обрабатывается на первом станке, а вторая — на втором. Требуется также, чтобы

$$\underline{t}_{i1} \leqslant \overline{t}_{i2},$$

где $\overline{t}_{i_1}$, $\underline{t}_{i_2}$ — как и прежде, соответственно время начала и окончания операции O_{ij}.

Отметим, что существует такое оптимальное решение, что последовательность обработки деталей на первом станке совпадает с последовательностью обработки деталей на втором станке.

Это свойство позволяет свести задачу к определению последовательности обрабатываемых деталей, т. е. к определению некоторой перестановки $\sigma = (i_1, i_2 \ldots, i_n)$ номеров 1, 2, ..., n, указывающей, в какой очередности следует обрабатывать детали.

Обозначим через σ_k последовательность $(i_1, i_2, \ldots, i_k)$ последовательности σ, через $A(\sigma_k)$ — время $t_{i_k 1}$, через $B(\sigma_k)$ — время $t_{i_k 2}$. Заметим, что

$$A(\sigma_{k+1}) = A(\sigma_k) + a_{i_{k+1}},$$
$$B(\sigma_{k+1}) = \max [A(\sigma_{k+1}),\ B(\sigma_k) + b_{i_{k+1}}].$$

Так как $\sigma_n = \sigma$, то

$$B(\sigma_n) = \max [B(\sigma_{n-1}),\ A(\sigma_n)] + b_{i_n} =$$
$$= \max [B(\sigma_{n-1}),\ A(\sigma_{n-1}) + a_{i_n}] + b_{i_n} =$$
$$= \max \{ \max [B(\sigma_{n-2}),\ A(\sigma_{n-2}) + a_{i_{n-1}}] + b_{i_{n-1}},\ A(\sigma_{n-2}) + a_{i_{n-1}} + a_{i_n}\} + b_{i_n}.$$

Очевидно, если изменить порядок обработки двух последних операций, то общая продолжительность обработки деталей может только увеличиться:

$$\max \{\max [B(\sigma_{n-2}),\ A(\sigma_{n-2}) + a_{i_{n-1}}] + b_{i_{n-1}};$$
$$A(\sigma_{n-2}) + a_{i_{n-1}} + a_{i_n}\} + b_{i_n} \leqslant$$
$$\leqslant \max \{ \max [B(\sigma_{n-2}),\ A(\sigma_{n-2}) + a_{i_n}] + b_{i_n};$$
$$A(\sigma_{n-2}) + a_{i_n} + a_{i_{n-1}}\} + b_{i_{n-1}}. \qquad (5.21)$$

Рассмотрим разность $B(\sigma_{n-2}) - A(\sigma_{n-2}) = b$.
Если положить, что $B(\sigma_{n-2}) = 0$ (перенесем начало отсчета времени), то неравенство (5.21) перейдет в неравенство

$$\max [\max (b,\ a_{i_{n-1}}) + b_{i_{n-1}};\ a_{i_{n-1}} + a_{i_n}] + b_{i_n} \leqslant$$
$$\leqslant \max [\max (b,\ a_{i_n}) + b_{i_n};\ a_{i_n} + a_{i_{n-1}}] + b_{i_{n-1}}. \qquad (5.22)$$

Произведя очередные преобразования, получим

$$\max [b,\ a_{i_{n-1}},\ a_{i_{n-1}} + a_{i_n} - b_{i_{n-1}}] \leqslant$$
$$\leqslant \max [b;\ a_{i_n};\ a_{i_{n-1}} + a_{i_n} - b_{i_n}]. \qquad (5.23)$$

Для выполнения условия (5.23) достаточно, чтобы

$$\max [a_{i_{n-1}} + a_{i_n} - b_{i_{n-1}};\ a_{i_{n-1}}] \leqslant \max [a_{i_n} + a_{i_{n-1}} - a_{i_n} \cdot a_{i_n}]$$

или

$$\max [-b_{i_{n-1}};\ -a_{i_n}] \leqslant \max [-b_{i_n}; -a_{i_{n-1}}],$$

что эквивалентно соотношению

$$\min [b_{i_{n-1}};\ a_{i_n}] \geqslant \min [b_{i_n};\ a_{i_{n-1}}]. \qquad (5.24)$$

Соотношение (5.24) будет выполняться в следующих случаях:

а) $a_{i_{n-1}} \leqslant b_{i_{n-1}}$; $a_{i_n} \leqslant b_{i_n}$; $a_{i_{n-1}} \leqslant a_{i_n}$;

б) $a_{i_{n-1}} \leqslant b_{i_{n-1}}$; $a_{i_n} \geqslant b_{i_n}$;

в) $a_{i_{n-1}} \geqslant b_{i_{n-1}}$; $a_{i_n} \geqslant b_{i_n}$; $b_{i_{n-1}} \geqslant b_{i_n}$.

Отсюда вытекает простой алгоритм определения оптимальной очередности обработки деталей в случае двух станков.

Найдем все детали, для которых $a_i \leqslant b_i$, и упорядочим их в порядке возрастания a_i. Детали, для которых $a_i > b_i$, упорядочим в порядке убывания b_i. Сначала следует обрабатывать детали первой группы, затем — второй.

Пример. Пусть характеристики деталей заданы таблицей

i	1	2	3	4	5	6
a_i	3	4	4	6	1	2
b_i	2	5	1	3	3	5

Очевидно, что для этих деталей оптимальной будет следующая очередность обработки деталей: 5, 6, 2, 4, 1, 3.

5.4. МЕТОД ВЕТВЕЙ И ГРАНИЦ

1. Общая схема

Одним из общих комбинаторных подходов к решению дискретных задач является метод ветвей и границ, впервые предложенный для решения задачи целочисленного линейного программирования [43]. В основе метода лежит, с одной стороны, процесс ветвления множества допустимых решений на дерево подмножеств, с другой стороны — процедура построения оценок подмножеств. Такой подход позволяет заменить полный перебор частичным. Изложим общую идею метода ветвей и границ [22].

Рассмотрим задачу дискретного программирования в следующей форме. Максимизировать

$$z = f(x) \tag{5.25}$$

при условии

$$x \in G. \tag{5.26}$$

Здесь G — некоторое конечное множество.

Метод ветвей и границ использует следующие процедуры.

Вычисление верхней границы. Часто удается найти верхнюю границу целевой функции f на множестве G (или на некотором

его подмножестве G'), т. е. такое число $\varsigma(G)(\varsigma(G'))$, что для $x \in G$ имеет место $f(x) \leqslant \varsigma(G)$ (соответственно для $x \in G'$ имеет место $f(x) \leqslant \varsigma(G')$).

Ветвление. 0-й шаг. Множество $G^0 = G$ разбивается на конечное число непересекающихся подмножеств

$$G'_1, G'_2, \ldots, G'_{t_1}.$$

***k*-й шаг ($k \geqslant 1$).** Имеются множества $G^k_1, G^k_2, \ldots, G^k_{t_k}$, еще не подвергавшиеся разбиению. По некоторому правилу среди них выбирается множество $G^k_{\mu(k)}$ и разбивается на конечное число непересекающихся подмножеств

$$G^k_{\mu(k),1}, G^k_{\mu(k),2}, \ldots, G^k_{\mu(k),s(k)}.$$

Неподвергавшиеся разбиению и вновь полученные множества

$$G^k_1, G^k_2, \ldots, G^k_{\mu(k)-1}, G^k_{\mu(k)+1}, \ldots, G^k_{t_k},$$

$$G^k_{\mu(k),1}, G^k_{\mu(k),2}, \ldots, G^k_{\mu(k),s(k)}$$

заново обозначаются через

$$G^{k+1}_1, G^{k+1}_2, \ldots, G^{k+1}_{t_{k+1}}.$$

Теорема 5.2. Пусть $\{G_1, G_2, \ldots, G_t\}$ разбиения множества G и $\bar{X}$ — план, принадлежащий некоторому подмножеству G_ν такой, что

$$f(\bar{X}) = \varsigma(G_\nu) \geqslant \varsigma(G_i), \quad i = 1, 2, \ldots, t.$$

Тогда $\bar{X}$ — оптимальный план задачи (5.25) — (5.26).

Доказательство непосредственно следует из определения верхней границы.

Изложим формальную схему метода ветвей и границ.

0-й шаг. Вычисляем оценку $\varsigma(G) = \varsigma(G^\circ)$.

Если при этом удается найти такой план $\bar{X}$, что

$$f(\bar{X}) = \varsigma(G),$$

то $\bar{X}$ — оптимальный план.

Если оптимальный план не найден , то по некоторому правилу разбиваем множество $G = G^\circ$ на конечное число непересекающихся подмножеств

$$G^\circ = G'_1 \cup G'_2 \cup \ldots \cup G'_{t_1}$$

и переходим к следующему шагу.

***k*-й шаг ($k \geqslant 1$).** Вычисляем оценки $\varsigma(G^k_i)$, $i = 1, 2, \ldots, t_k$. Если при этом удается найти такой план $\bar{X}$, что $\bar{X} \in G^k_t$ для некоторого t $(1 \leqslant t \leqslant t_k)$ и

$$f(\bar{X}) = \varsigma(G^k_t) \geqslant \varsigma(G^k_i), \quad i = 1, 2, \ldots, t_k,$$

то $\bar{X}$ — оптимальный план.

Если оптимальный план не найден, то выбираем наиболее перспективное множество $G^k_{\mu(k)}$ по правилу

$$\varsigma(G^k_{\mu(k)}) = \max_{i=1, 2, \ldots, t_k} \varsigma(G^k_i).$$

Разбиваем $G^k_{\mu(k)}$ на несколько непересекающихся подмножеств

$$G^k_{\mu(k)} = G^k_{\mu(k),1} \cup G^k_{\mu(k),2} \cup \ldots \cup G^k_{\mu(k),s(k)}.$$

Еще не подвергавшиеся разбиению и вновь полученные множества

$$G^k_1,\ G^k_2,\ \ldots,\ G^k_{\mu(k)-1},\ G^k_{\mu(k)+1} \ldots G^k_{t_k},$$

$$G^k_{\mu(k),1},\ G^k_{\mu(k),2},\ \ldots,\ G^k_{\mu(k),s(k)}$$

заново обозначаем через

$$G^{k+1}_1,\ G^{k+1}_2,\ \ldots, G^{k+1}_{t_{k+1}}$$

и переходим к $(k+1)$-му шагу.

Для реализации описанной выше схемы применительно к конкретным задачам необходимо конкретизировать правила ветвления, вычисления оценок и нахождения планов. Покажем, как это можно сделать для целочисленной задачи линейного программирования с переменными 0—1.

2. Задача целочисленного программирования

Рассмотрим задачу линейного программирования, состоящую в максимизации функционала

$$L(X) = \sum_{j=1}^{n} c_j x_j \tag{5.27}$$

при условиях

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i,\ i = 1,\ 2,\ \ldots,\ m; \tag{5.28}$$

$$0 \leqslant x_j \leqslant 1,\quad j = 1,\ 2,\ \ldots,\ n; \tag{5.29}$$

$$x_j \text{ — целые},\ j = 1,\ 2,\ \ldots,\ n. \tag{5.30}$$

Будем предполагать, что $c_j \geqslant 0$, $j=1, 2, \ldots, n$. Этого всегда можно добиться заменой переменных

$$\bar{x}_j = 1 - x_j \quad \text{, если } c_j < 0,$$

$$\bar{x}_j = x_j \quad \text{, если } c_j \geqslant 0.$$

Через M и N обозначим соответственно множества индексов $\{1, 2, \ldots, m\}$ и $\{1, 2, \ldots, n\}$. Множество планов задачи обозначим через G.

Покажем, каким образом можно найти достаточно точную верхнюю границу $\varsigma(G)$ значений формы (5.27) при условиях (5.28) — (5.30).

Рассмотрим задачу A_i максимизации (5.27) при условиях (5.29) и

$$\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i.$$

Обозначим через I_i^+ множество индексов j, для которых $a_{ij} > 0$, а через I_i^- множество индексов j, для которых $a_{ij} \leqslant 0$. Очевидно, если существует решение $(x_1^0, x_2^0, \ldots, x_n^0)$ задачи A_i, то $x_j^0 = 1$, для $j \in I_1^-$. Следовательно, задача A_i сводится к задаче A_i^* максимизации формы

$$h_i + \sum_{j \in I_i^+} c x_j$$

при условиях (5.29) и

$$\sum_{j \in I_i^+} a_{ij} x_j \leqslant b_i^*.$$

Здесь

$$a_{ij} > 0 \,(j \in I_i^+),\ b_i^* = b_i - \sum_{l \in I_i^-} - a_{ij},\ h_i = \sum_{j \in I_i^-} c_j.$$

Будем предполагать, что $b_i^* \geqslant 0$, так как в противном случае задача A_i, а следовательно, и задача (5.27) — (5.30) не имеет решений.

Утверждение 1. Пусть все переменные x_j задачи A_i^* перенумерованы так, что

$$\frac{c_1}{a_{i1}} \geqslant \frac{c_2}{a_{i2}} \geqslant \ldots \geqslant \frac{c_k}{a_{ik}},\ k = |I_i^+| \leqslant n.$$

Тогда, если $r_0 = \max \{r \,|\, \sum_{j \leqslant r} a_{ij} \leqslant b_i^*\}$, то решение задачи A_i^* получается по формуле

$$x_j = \begin{cases} 1, & \text{если } j \leqslant r_0, \\ 0, & \text{если } j > r_0 + 1, \end{cases}$$

$$x_{r_0+1} = (b_i^* - \sum_{j \leqslant r_0} a_{ij}) : a_{i,\, r_0+1},$$

т. е. максимальное значение L_i функционала задачи A_i будет равно

$$L_i = h_i + \sum_{j \leqslant r_0} c_j + \frac{b_i^* - \sum_{j \leqslant r_0} a_{ij}}{a_{i,\, r_0+1}} c_{r_0+1}.$$

Отсюда непосредственно следует, что число $\varsigma(G) = \min_{i \in M} L_i$ является верхней границей значений формы (5.27) при условиях (5.28) — (5.30), т. е. $\varsigma(G) \geqslant L(x)$ для любого плана задачи (5.27)—(5.30).

Аналогично определяется число $\varsigma(G^k_\mu)$, т. е. верхняя граница линейной формы задачи (5.27) — (5.30) с дополнительными условиями

$$x_j = 1, \quad j \in v^k_\mu, \qquad (5.31)$$
$$x_j = 0, \quad j \in u^k_\mu,$$

где

$$u^k_\mu,\ v^k_\mu \subset N,\ u^k_\mu \cap v^k_\mu = \varnothing.$$

В процессе ветвления множество к $G^k_{\mu(k)}$ разбивается на два множества:

$$G^k_{\mu(k)} = G^k_{\mu(k),\,1} \cup G^k_{\mu(k),2},$$

причем

$$u^k_{\mu(k),\,1} = u^k_{\mu(k)} \cup \{|u^k_{\mu(k)}| + |v^k_{\mu(k)}| + 1\},$$
$$v^k_{\mu(k),\,1} = v^k_{\mu(k)},$$
$$u^k_{\mu(k),\,2} = u^k_{\mu(k)},$$
$$v^k_{\mu(k),\,2} = v^k_{\mu(k)} \cup \{|u^k_{\mu(k)}| + |v^k_{\mu(k)}| + 1\}.$$

Через $X^k_{\mu(k)}$ обозначим псевдоплан, соответствующий множеству $G^k_{\mu(k)}$,

$$X^k_{\mu(k)} = \{x_j \mid x_j = 1,\ j \in v^k_{\mu(k)},\quad x_j = 0,\ j \in N \setminus v^k_{\mu(k)}\}.$$

Алгоритм решения задачи (5.27) — (5.30) состоит в следующем.

0-й шаг. Вычисляем оценку $\varsigma(G) = \varsigma(G_0)$ $(u_0 = \varnothing,\ v_0 = \varnothing)$. Если псевдоплан X_0 является планом задачи (5.27) — (5.30) и $\varsigma(G_0) = L(X_0)$, то X_0 — оптимальный план.

Если оптимальный план не найден, разбиваем множество на непересекающиеся подмножества

$$G_0 = G^1_1 \cup G^2_1.$$

Среди псевдопланов $\overline{X}^1_1$, X^2_1, являющихся планами исходной задачи, выбираем план $\overline{X}_1$ с максимальным значением функционала и переходим к следующему шагу.

***k*-й шаг ($k = 1, 2, \ldots$).** Вычисляем оценки $\varsigma(G^k_i)$, $k = 1, 2, \ldots, t_k$. Если при этом $L(\overline{X}_k) \geqslant \varsigma(G^k_i)$, $i = 1, 2, \ldots, t_k$, то $\overline{X}_k$ — оптимальный план. Если оптимальный план не найден, то выбираем множество $G^k_{\mu(k)}$ по правилу

$$\varsigma(G^k_{\mu(k)}) = \max_{i=1,\,2,\,\ldots,\,t_k} \varsigma(G^k_i).$$

Разбиваем $G^k_{\mu(k)}$ на два непересекающихся подмножества по вышеизложенному правилу. Еще неподвергавшиеся разбиению и вновь полученные множества заново обозначаем через

$$G^{k+1}_1,\ G^{k+1}_2,\ \ldots,\ G^{k+1}_{t_{k+1}}.$$

Среди псевдопланов X_1^{k+1}, X_2^{k+1}, . . . , $X_{t_{k+1}}^{k+1}$, являющихся планами исходной задачи, выбираем план $\overline{X}_{k+1}$ с максимальным значением функционала и переходим к $(k+1)$-му шагу.

Проиллюстрируем работу алгоритма на примере.

Пример. Максимизировать

$$L(X) = 5x_1 + 2x_2 + 4x_3,$$

при условиях

$$2x_1 + 2x_2 + x_3 \leqslant 4,$$
$$-2x_1 + 3x_2 + 3x_3 \leqslant 3,$$
$$2x_1 - 2x_2 + x_3 \leqslant 1,$$
$$x_j = 0 \text{ или } 1,\ j = 1,\ 2,\ 3.$$

0-й шаг. $\varsigma(G_0) = \min\{11,\ 9,\ 11\} = 9$.

Так как $X_0 = (0,\ 0,\ 0)$ — план задачи и $\varsigma(G_0) = 9 > L(X_0) = 0$, то разбиваем множество G_0 на подмножества G_1^1, G_2^1, причем $u_1^1 = \{1\}$, $v_1^1 = \varnothing$, $u_2^1 = \varnothing$, $v_2^1 = \{1\}$. Псевдопланы $X_1^1 = (0,\ 0,\ 0)$ и $X_2^1 = \{1,\ 0,\ 0\}$ удовлетворяют условиям задачи. Так как $L(X_1^1) = 0$ и $L(X_2^1) = 5$, то полагаем $\overline{X}_1 = X_2^1$ и переходим к следующему шагу.

1-й шаг. Вычисляем оценки $\varsigma(G_1^1) = 4$, $\varsigma(G_2^1) = 7$. Так как $L(\overline{X}_1) = 5 < \varsigma(G_2^1) = 7$, то $\overline{X}_1$ не является оптимальным планом. Для разбиения выбираем множество G_2^1, так как $\varsigma(G_2^1) > \varsigma(G_1^1)$. Производим разбиение множества G_2^1:

$$G_2^1 = G_{2,1}^2 \cup G_{2,2}^2,$$

причем

$$u_{2,1}^1 = \{2\},\ v_{2,1}^1 = \{1\},\ u_{2,2}^1 = \varnothing,\quad v_{2,2}^1 = \{1,\ 2\}.$$

Переобозначаем неподвергавшиеся разбиению множества

$$G_1^1 = G_1^3,\ G_{2,1}^2 = G_2^3,\ G_{2,2}^2 = G_3^3.$$

Псевдопланы $X_2^3 = (100)$ и $X_3^3 = (11,\ 0)$ удовлетворяют условиям задачи. Полагаем $\overline{X}_2 = X_3^3$, так как $L(X_3^3) = 7$,

$$L(X_2^3) = 5,\ \text{а}\ L(X_1^3) = 0.$$

2-й шаг. Вычисляем оценки $\varsigma(G_1^3) = 4$, $\varsigma(G_2^3) = 5$, $\varsigma(G_3^3) = 7$.

Так как $L(\overline{X}_2) = 7 = \varsigma(G_3^3) = 7$, то $\overline{X}_2$ — оптимальный план задачи.

Итак, в оптимальном плане задачи

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0.$$

5.5. МЕТОД ПОСТРОЕНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ПЛАНОВ

1. Общая схема

Метод построения последовательности планов использовался в последнее время для решения целочисленных задач линейного программирования [21], задач дискретного программирования [12], задач размещения производства [13] и др. Изложим сущность метода построения последовательности планов на примере задачи дискретного программирования.

Рассмотрим задачу A минимизации функционала

$$F(X) = g_1(x_1)\,\omega\, g_2(x_2)\,\omega \ldots \omega\, g_m(x_m)\,\omega\, Q(x_1,\ x_2, \ldots,\ x_m)$$

при ограничениях

$$\sum_{i=1}^{m} a_i x_i \geqslant c; \tag{5.32}$$

$$X = (x_1,\ x_2,\ \ldots,\ x_m) \in G, \tag{5.33}$$

$$x_i \in \gamma_i = \{a_{i1},\ a_{i2},\ \ldots,\ a_{is_i}\},\ i = 1,\ 2,\ \ldots,\ m, \tag{5.34}$$

где a_i, c, a_{ij} — целые положительные числа; ω — операция сложения или умножения; $Q(X)$ — функция, для которой существуют такие функции $t_i(x_i)$, $i = 1, 2, \ldots, m$, что

$$t_1(x_1)\,\omega\, t_2(x_2)\,\omega \ldots \omega\, t_m(x_m) \leqslant Q(X)$$

для любого плана задачи A. Множество G может быть задано различными способами (уравнениями, неравенствами, логическими условиями и т. д.). Множество планов задачи A обозначим через M, а множество планов, определенное ограничениями (5.32) и (5.34), через $\overline{M}$.

Излагаемый метод решения задачи A состоит в построении и анализе последовательности планов вспомогательной задачи $\overline{A}$ минимизации функционала $\overline{F}(X) = \overline{g}_1(x_1)\,\omega\,\overline{g}_2(x_2)\,\omega\ \ldots\ \omega\,\overline{g}_m(x_m)$ на множестве $\overline{M}$. Здесь $\overline{g}_i(x_i) = g_i(x_i)\,\omega\, t_i(x_i)$, $i = 1, 2, \ldots, m$.

Пусть $f_h(z)$ — оптимальное значение функционала задачи $\overline{A}$, в которой индекс m и параметр c заменены соответственно на h и z. План $X_0 = (x_1^0,\ x_2^0,\ \ldots,\ x_m^0)$ задачи $\overline{A}$ назовем h-оптимальным ($1 \leqslant h \leqslant m$) и обозначим через

$$(*,\ *,\ \ldots,\ *,\ x_{h+1}^0,\ \ldots,\ x_m^0),$$

если

$$\overline{F}(X_0) = \overline{g}_{h+1}(x_{h+1}^0)\,\omega\ \ldots\ \omega\,\overline{g}_m(x_m^0)\,\omega\, f_h\left(c - \sum_{i=h+1}^{m} a_i x_i^0\right).$$

При этом множество планов задачи $\overline{A}$ таких, что $x_i = x_i^0$, $i = h+1, \ldots, m$ обозначим через $\overline{M}(*,\ *,\ \ldots,\ *,\ x_{h+1}^0,\ \ldots,\ x_m^0) = \overline{M}(X_0)$.

Непосредственно из определения h — оптимальности плана X_0 вытекает.

Лемма 5.1. $\overline{F}(X_0) = \min\limits_{x \in \overline{M}(X_0)} F(X)$

Опишем алгоритм φ построения последовательности планов задачи $\overline{A}$.

1-й шаг. Среди множества W_1 всех $(m-1)$-оптимальных планов задачи $\overline{A}$ находим такой план

$$X_1 = (*,\ *,\ \ldots,\ *,\ x_m^1) = (x_1^1,\ x_2^1,\ \ldots,\ x_m^1),$$

что

$$F(X_1) = \min_{x \in W_1} \overline{F}(X).$$

k-й шаг ($k = 2, 3, \ldots$). Множество W_{k-1} преобразуем во множество W_k по следующему правилу. Оставляем без изменения все планы множества W_{k-1}, кроме плана

$$X_{k-1} = (*, *, \ldots, *, x_p^{k-1}, \ldots, x_m^{k-1}) = (x_1^{k-1}, x_2^{k-1}, \ldots, x_m^{k-1}),$$

который заменяем системой всевозможных планов вида

$$(*, *, \ldots, *, x_{q-1}, x_q^{k-1}, \ldots, x_p^{k-1}, \ldots, x_m^{k-1}),$$

где

$$x_{q-1} \in \gamma_{q-1}, \quad x_{q-1} \neq x_{q-1}^{k-1}, \quad q = 2, 3, \ldots, p,$$

причем

$$\sum_{i=1}^{q-2} \max_{1 \leqslant k \leqslant S_i} a_{ik} \geqslant c - \sum_{i=q}^{m} x_i^{k-1} - x_{q-1}.$$

Далее находим такой план X_k, что $\overline{F}(X_k) = \min_{x \in W_k} \overline{F}(X)$ и переходим к следующему $(k+1)$-му шагу.

Введем обозначение $\overline{M}_k = \bigcup_{X \in W_k} \overline{M}(X)$.

Лемма 5.2. $\overline{M}_1 = \overline{M}$, $\overline{M}_k = \overline{M}_{k-1} \setminus X_{k-1}$, $k = 2, 3, \ldots$

Из лемм 5.1 и 5.2 вытекает

Теорема 5.3. Алгоритм φ строит такую последовательность планов $X_1, X_2, \ldots$ задачи $\overline{A}$, что

$$\overline{F}(X_k) = \min_{X \in M \setminus \{x_1, x_2, \ldots, x_{k-1}\}} \overline{F}(X), \quad k = 1, 2, \ldots$$

Иначе говоря, алгоритм φ строит последовательность планов задачи $\overline{A}$ в порядке неубывания функции $F(X)$.

Критерий оптимальности. Если существует такое натуральное число k, что $M_k = \{X_1, X_2, \ldots, X_k\} \cap M \neq \varnothing$ и

$$\overline{F}(X_k) \geqslant \min_{X \in M_k} F(X) = F(X^*), \tag{5.35}$$

то X^* — оптимальный план задачи A.

Критерием оптимальности указывается алгоритм ψ решения задачи A. Искомый алгоритм ψ описывается следующим образом: k-й шаг ($k = 1, 2, \ldots$) состоит в том, что с помощью алгоритма φ строим план X_k и проверяем выполнение критерия оптимальности. Если критерий оптимальности выполняется, то процесс обрывается и X^* — оптимальный план задачи A. Если нет, то переходим к следующему шагу.

2. Задача дискретного программирования

Рассмотрим применение метода построения последовательности планов к решению одной часто встречающейся на практи-

ке задачи дискретного программирования. Задача состоит в минимизации функционала

$$F(X) = F(X_1,\ X_2,\ \ldots,\ X_t) = \sum_{i \in I} g_i(x_i) \qquad (5.36)$$

при ограничениях

$$\sum_{i \in I_t} x_i = c_t,\ t = 1,\ 2,\ \ldots,\ v, \qquad (5.37)$$

$$\sum_{i \in I} \varphi_{ik}(x_i) \leqslant r_k,\ k = 1,\ 2,\ \ldots,\ l, \qquad (5.38)$$

$$x_i \in \gamma_i = \{a_{i1},\ \ldots,\ a_{is_i}\};\ i \in I, \qquad (5.39)$$

где $I_t \subset I$, $t = 1,\ 2,\ \ldots,\ v$, $I_k \cap I_l = \varnothing$ при $k \neq l$, $\bigcup_{i=1}^{v} I_t = I$.

Множество планов задачи (5.36) — (5.39) обозначим через G. В качестве вспомогательной задачи возьмем задачу A минимизации функционала $F(X)$ при условиях (5.37) и (5.39).

Очевидно, задача A имеет решение, если разрешима каждая из задач A_t, $t=1, 2, ..., v$, состоящая в минимизации функционала

$$F_t(X_t) = \sum_{i \in I_t} g_i(x_i).$$

при ограничениях $\sum_{i \in I_t} x_i = c_t,$

$$x_i \in \gamma_i,\ i \in I_t.$$

В дальнейшем будем предполагать, что каждая из задач A_t, $t=1, 2, ..., v$ имеет решение.

Ранее было показано (алгоритм φ), как можно строить последовательность планов $X_t^1, X_t^2, \ldots$ задачи A_t таких, что

$$F_t(X_t) = \min_{X \in M_t^k} F_t(X),\ k = 1,\ 2,\ \ldots,$$

где $M_t^k = M_t^{k-1} \setminus X_t^{k-1}, k = 2,\ 3,\ \ldots,\ M_t^1$—множество планов задачи A_t.

Очевидно, план $\beta_1 = (X_1^1,\ X_2^1,\ \ldots,\ X_v^1)$ задачи A является оптимальным, т. е.

$$F(\beta_1) = \min_{\beta \in M_1} F(\beta),$$

где M_1 — множество планов задачи A.

Для произвольного плана $\beta = (X_1, X_2, \ldots, X_{p-1}, X_p^q, X_{p+1}^1, \ldots, X_v^1)$, $(q \neq |M_p^1|,\ 1 \leqslant p \leqslant v)$ задачи A определим $O(\beta)$ следующим образом:

$$O(\beta) = \{\beta^{(0)},\ \beta^{(1)},\ \ldots,\ \beta^{(v-p)}\}, \text{ где}$$

$$\beta^{(0)} = (X_1,\ X_2,\ \ldots,\ X_{p-1},\ X_p^{q-1},\ X_{p+1}^1,\ \ldots,\ X_v^1),$$

$$\beta^{(s)} = (X_1,\ X_2,\ \ldots,\ X_{p-1},\ X_p^q,\ X_{p+1}^1,\ \ldots,\ X_{p+s}^2,\ X_{p+s+1}^1, \ldots, X_v^1),\ s = 1,\ 2,\ \ldots,\ v-p.$$

Очевидно, если $p=v$ и $g=|M_v^1|$, то $O(\beta)=\varnothing$.

Кроме того, $O(\beta_1)=\{\beta^{(0)}, \beta^{(1)}, \ldots, \beta^{(v-1)}\}$,

где $\beta(s)=(X_1^1, X_2^1, \ldots, X_s^1, X_{s+1}^2, X_{s+2}^1, \ldots, X_v^1)$, $s=0,1,\ldots,v-1$.

Опишем алгоритм φ построения последовательности планов задачи A.

1-й шаг. Строим план $\beta_1=(X_1^1, X_2^1, \ldots, X_v$ и полагаем $W_1=\{\beta_1\}$.

***k*-й шаг** ($k=2, 3, \ldots$). Строим множество планов

$$W_k=\bigcup_{s=1}^{k-1} O(\beta_s)\setminus\{\beta_1, \beta_2, \ldots, \beta_{k-1}\}.$$

Находим план β_k такой, что

$$F(\beta_k)=\min_{\beta\in W_k} F(\beta),$$

и переходим к следующему шагу.

Нетрудно показать, что последовательность планов $\beta_1, \beta_2, \ldots$ такова, что

$$F(\beta_k)=\min_{\beta\in M_k} F(\beta),\ k=1, 2, \ldots,$$

где $M_k=M_{k-1}\setminus\beta_{k-1}$, $k=2, 3, \ldots$.

Итак, алгоритм φ строит последовательность планов задачи A в порядке неубывания функционала $F(\beta)$.

Аналогично [12] справедлива

Теорема 5.4. План β_k последовательности $\beta_1, \beta_2 \ldots$ такой, что $\beta_k\in G$, а $\beta_l\notin G$, $l=1, 2, \ldots, k-1$ является, оптимальным планом задачи (5.36) — (5.39).

Непосредственно из теоремы 5.4 вытекает следующий алгоритм ψ поиска оптимального плана задачи (5.36) — (5.39).

На k-м шаге ($k=1, 2, \ldots$) строим план β_k задачи A и проверяем, удовлетворяет ли он условиям (5.38). Если да, то β_k — оптимальный план задачи (5.36) — (5.39). Если нет, то переходим к $(k+1)$-му шагу.

Принципиальная блок-схема этого алгоритма приведена на рис. 5.4.

Пример. Минимизировать функционал

$$F(X_1, X_2)=\sum_{i\in I} g_i(x_i). \tag{5.36'}$$

при условиях

$$\sum_{i\in I_1} x_i=12,\ \sum_{i\in I_2} x_i=16, \tag{5.37'}$$

$$\sum_{i\in I} \varphi_i(x_i)\leqslant 24, \tag{5.38'}$$

$$x_i\in\gamma_i,\ i\in I, \tag{5.39'}$$

где $I=\{1, 2, 3, 4, 5, 6,\}$, $I_1=\{1, 2, 3\}$, $I_2=\{4, 5, 6\}$,

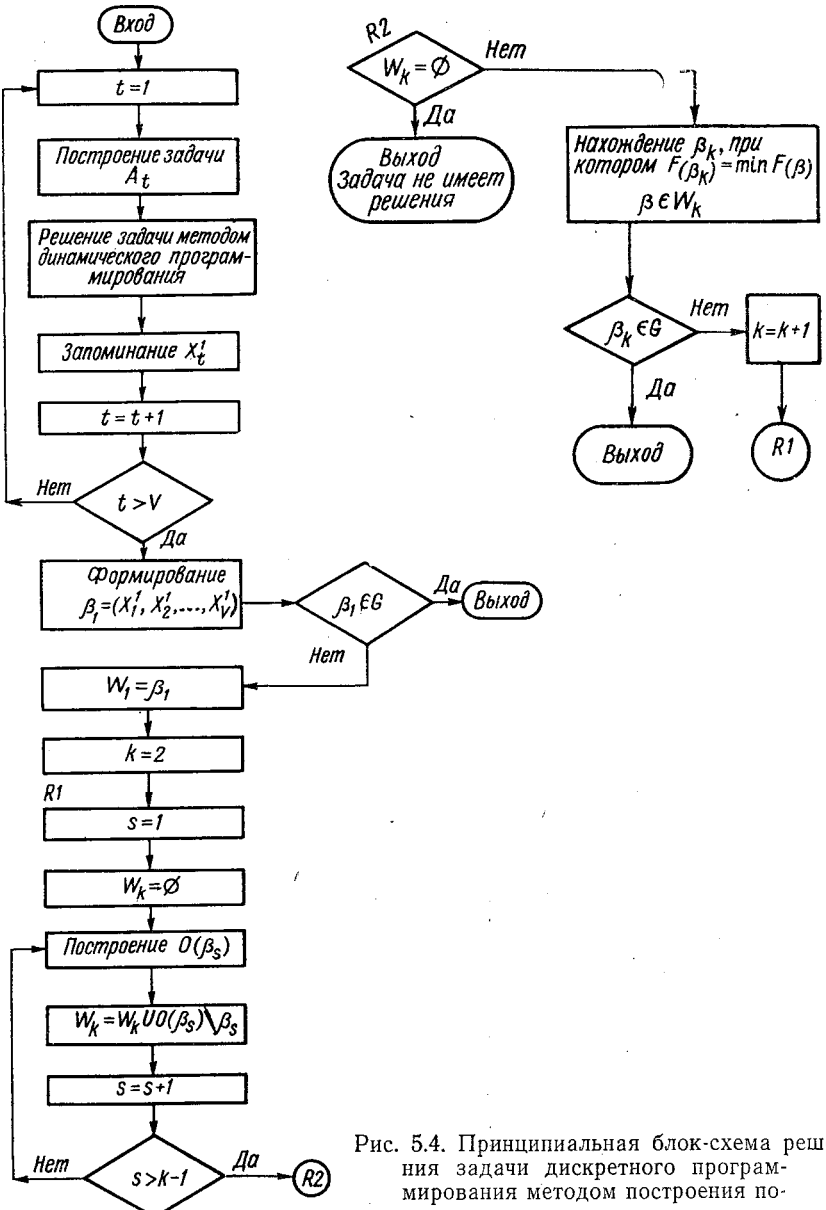


Рис. 5.4. Принципиальная блок-схема решения задачи дискретного программирования методом построения последовательности планов

$\gamma_1 = \{0, 3, 5\}$, $\gamma_2 = \{0, 2, 4\}$, $\gamma_3 = \{0, 3, 5\}$,
$\gamma_4 = \{0, 6\}$, $\gamma_5 = \{0, 6, 10\}$ $\gamma_6 = \{0, 4, 10\}$,

а функции $g_i(x_i)$, $\varphi_i(x_i)$, $i \in I$, $x_i \in \gamma_i$ заданы таблицами 1 и 2.

Таблица 1

x	0	2	3	4	5
$g_1(x)$	0	—	26	—	40
$\varphi_1(x)$	0	—	4	—	6
$g_2(x)$	0	16	—	42	—
$\varphi_2(x)$	0	3	—	5	—
$g_3(x)$	0	—	11	—	25
$\varphi_3(x)$	0	—	4	—	6

Таблица 2

x	0	4	6	10
$g_4(x)$	0	—	28	—
$\varphi_4(x)$	0	—	3	—
$g_5(x)$	0	—	38	64
$\varphi_5(x)$	0	—	3	5
$g_6(x)$	0	9	—	45
$\varphi_6(x)$	0	2	—	5

Р е ш е н и е. 1-й шаг. Решая задачи A_1^1 и A_2^1, получаем $X_1^1=(5, 2, 5)$ и $X_2^1=(6, 6, 4)$.

Следовательно, $\beta_1 = (X_1^1, X_2^1)$ — решение задачи A, т. е. в данном случае задачи (5.36′), (5.37′), (5.39′). Причем $F(X_1^1, X_2^1) = 136$. План β_1 не удовлетворяет условию (5.38′).

Полагаем $W_1 = \{\beta_1\}$ и переходим ко второму шагу.

2-й шаг. Строим множество планов $W_2 = O(\beta_1) \setminus \beta_1$. Так как $O(\beta_1) = \{\beta^{(0)}, \beta^{(1)}\}$, где $\beta^{(0)} = \{X_1^2, X_2^1\}$, $\beta^{(1)} = \{X_1^1, X_2^1\}$, $X_1^2 = (3, 4, 5)$,

$$X_2^2 = (0, 6, 10), \text{ то } W_2 = \{(X_1^2, X_2^1), (X_1^1, X_2^2)\}.$$

Находим план β_2. В нашем случае $\beta_2 = (X_1^1, X_2^2)$, причем $F(\beta_2) = 164$.

План β_2 удовлетворяет условию (5.38′), т. е. является решением исходной задачи.

Глава 6

ЭТАПЫ ПРОЕКТИРОВАНИЯ СИСТЕМЫ ОБРАБОТКИ ДАННЫХ В АСУП

6.1. ОБЩИЕ СВЕДЕНИЯ

Разработка и внедрение АСУП — это сложный и длительный процесс, включающий следующие этапы [27]: предпроектное обследование объекта управления (результатом обследования является техническое задание на разработку АСУП); разработку технического проекта (одним из основных результатов этого этапа является комплекс постановок задач, решение которых предусматривается на данном объекте); разработку рабочего проекта, включающего программы для решения задач, соответствующие инструкции персоналу, участвующему в функционировании АСУП, установку и введение в действие необходимых средств вычислительной техники, внедрение новой системы документооборота на объекте; опытную эксплуатацию системы, при которой проверяется качество рабочего проекта, проводятся некоторые модернизации его и осуществляется последовательный переход на машинную обработку данных. При полном переходе на машинную обработку данных начинается этап промышленной эксплуатации АСУП. При этом, естественно, могут продолжаться работы по совершенствованию АСУП, по расширению круга решаемых задач, по совершенствованию внедренных задач, по улучшению принятой системы документооборота. Как следствие разработанный проект может корректироваться в результате анализа функционирования АСУП, а это в свою очередь накладывает существенное требование на систему математического обеспечения АСУП, заключающееся в возможности простой корректировки программ в зависимости от создавшихся условий.

Одной из наиболее важных и трудоемких частей разработки АСУП является создание комплекса информационно-увязанных задач, который принято называть системой электронной обработки данных (СЭОД). Рассмотрим основные этапы создания СЭОД, включающие разработку постановок задач и принципиальных схем решения задач, проведение информационной увязки задач, разработку блок-схем и программ, отладку программ,

проведение опытной и промышленной эксплуатации системы. Характеристика всех этапов будет сопровождаться комплексным примером задачи «Расчет плана поставок готовой продукции».

6.2. ПОСТАНОВКА ЗАДАЧИ

Постановка задачи разрабатывается в основном организаторами производства и специалистами по экономическим вопросам. В ней разъясняется организационно-экономическая сущность задачи и содержится описание входной оперативной, нормативно-справочной, выходной, накапливаемой и промежуточной информации, описание информации по внесению изменений в массивы, алгоритм решения задачи и контрольный пример.

Организационно-экономическую сущность задачи составляют наименование, краткое содержание, периодичность решения, связь данной задачи с другими и ее место в комплексе задач СЭОД, способ организации сбора необходимых данных и передачи их на обработку с указанием используемых при этом периферийных технических средств и носителей информации, временные ограничения на получение результатов решения, специфические особенности этой задачи.

В качестве носителей входной оперативной информации могут быть документы, перфокарты или перфоленты.

По каждой форме документа приводятся шифр, периодичность и ориентировочные сроки поступления, максимальное количество документострок, перфорируемые реквизиты и способ обнаружения ошибок.

В графостроке формы по каждому реквизиту дается его характеристика, включающая условное обозначение и диапазон изменения значения этого реквизита.

Указанные характеристики записываются в виде отдельной таблицы или непосредственно на форме документа.

Если в качестве носителей информации взяты перфоленты или перфокарты, получаемые на периферийных технических средствах, то для них приводятся соответствующие макеты перфорации. Описание этих данных осуществляется по тем же правилам, что и для документов.

Нормативно-справочная информация может быть представлена в виде форм первичных документов или в виде массивов, записанных на магнитные ленты. В последнем случае необходимо указать структуру этих массивов, включающую соответствующие описания записей и реквизитов. Кроме того, в описании организационно-экономической сущности задачи должны быть приведены необходимые сведения по системе шифровки реквизитов (в той ее части, которая используется в алгоритме решения данной задачи), сроки хранения нормативных данных, указание на необходимость введения дополнительных реквизитов, если это требуется для решения других задач.

При описании выходной информации приводятся формы выходных документов и дополнительные сведения, включающие периодичность получения соответствующей формы документа, количество экземпляров, среднее количество строк в документе, требования к последовательности получения документов на ЭВМ, порядок рассортированности строк, характеристику реквизитов в выходном документе, специальные требования к оформлению документов (например, необходимость разбиения документа на страницы и количество строк на странице, подписи должностных лиц, необходимость выделения отдельных строк и др.).

К накапливаемой информации относятся данные, являющиеся результатами решения задачи и служащие в качестве исходной информации для ее последующего решения. Промежуточная информация также образуется в результате решения данной задачи, но служит в качестве исходных данных для решения других задач. Описание этих массивов данных состоит в перечислении состава реквизитов и их форматов.

При описании информации по внесению изменений приводятся перечень наименований реквизитов, значения которых подвергаются изменениям, а также перечень и характер разновидностей этих изменений, среднее и максимальное количество документострок при каждом внесении изменений.

Алгоритм решения задачи содержит расчетные формулы, определяющие соотношения между входными, промежуточными и выходными для данной задачи реквизитами; указание на режим выполнения отдельных частей алгоритма в зависимости от заданных условий; контрольные соотношения, используемые для контроля результатов; и действия, которые должны быть выполнены, если контрольные соотношения нарушены.

Контрольный пример служит для проверки правильности алгоритмов решения задачи и отладки программ.

Контрольный пример должен содержать входные документы с оперативной, нормативной и используемой хранимой информацией, заполненные конкретными данными; результаты решения задачи, хранимые для других задач и накапливаемые для решения данной задачи; выходные документы, содержащие данные, полученные в соответствии с заданным алгоритмом.

Данные контрольного примера должны содержать сведения, охватывающие основные варианты алгоритма и обеспечивающие полную проверку алгоритма решения задачи.

Рассмотрим постановку задачи «Расчет плана поставок готовой продукции».

Организационно-экономическая сущность задачи. Задача «Расчет плана поставок готовой продукции» предназначена для финансового, производственно-диспетчерского, планово-экономического отделов и отдела сбыта часового завода. Цель задачи — на основании договорных спецификаций, разнарядок и заказ-нарядов разработать годовой, квартальный и месячный

планы поставок готовой продукции потребителям и согласовать их с планом производства.

В результате решения этой задачи необходимо получить следующие документы:

«План поставок продукции на год» (форма ПГП1);

«План поставок продукции по потребителям на квартал (месяц)» (форма ПГП2);

«Соответствие плана производства портфелю заказов на квартал (месяц)» (форма ПГП3);

«Ведомость соответствия разнарядок спецификациям» (форма ПГП4);

«Ведомость отклонений разнарядок от спецификаций» (форма ПГП5).

Документ ПГП1 предназначен для планово-экономического отдела завода и служит основой для составления годового плана производства в разрезе шифров изделий. Данный документ должен быть рассчитан к началу каждого года.

Документ ПГП2 должен быть получен к началу каждого квартала на основании поступивших на предприятие разнарядок на поставку продукции. По данной форме работники отдела сбыта получают как квартальные, так и месячные планы поставок продукции в разрезе шифров покупателей, грузополучателей и шифров изделий.

Документ ПГП3 дает возможность предприятию проконтролировать соответствие плана производства поступившим заказам по разнарядкам по каждому шифру изделий.

К началу каждого квартала или по мере необходимости определяется соответствие квартального плана поставок поступившим разнарядкам на получение готовой продукции, что находит отражение в выходном документе ПГП4, а отклонения по шифрам изделий в разрезе шифров покупателей выдаются в виде документа ПГП5.

При решении задачи в качестве входной оперативной информации используются следующие документы:

«План производства продукции» (форма ПГП6);

«Разнарядка на поставку продукции» (форма ПГП7);

«Заказ-наряд о поставке продукции на экспорт» (форма ПГП8).

В качестве нормативно-справочной информации используются следующие справочники: «Справочник потребителей продукции» (форма НСИ1), «Справочник цен по изделиям» (форма НСИ2), «Спецификация на поставку продукции» (форма НСИ3).

Задача «Расчет плана поставок готовой продукции» может решаться или локально, или в комплексе с другими задачами подсистемы сбыта и реализации продукции.

Результаты решения данной задачи используются в качестве исходной информации для решения следующих задач: «Оперативный учет выполнения плана отгрузки продукции», «Состав-

План производства продукции на _____ квартал _____ года

Шифр изделия	Количество изделий на квартал	В том числе по месяцам		
		1	2	3
1	2	3	4	5
9(11)	9(7)	9(7)	9(7)	9(7)

Рис. 6.1. Форма входного документа ПГП6

Разнарядка на поставку продукции **А (40)** (наименование фондодержателя) **Шифр фондодержателя 9(4)**

Шифр грузополучателя	Наименование грузополучателя	Адрес грузополучателя	Процент торговой скидки	Шифр плательщика	Наименование и адрес плательщика	Местонахождение банка	Номер расчетного счета	Шифр изделия	Количество на квартал	В том числе по месяцам (шт.)		
										1	2	3
1	2	3	4	5	6	7	8	9	10	11	12	13
9(4)	А(20)	А(40)	9(2), 9(2)	9(4)	А(60)	А(72)	9(9)	9(11)	9(7)	9(7)	9(7)	9(7)

Рис. 6.2. Форма входного документа ПГП7

Внешнеторговое
объединение 9(2)

Кому _____

Заказ-наряд на поставку продукции на экспорт № 9(13)

На основании _____ заказываем

Шифр товара (изделия)	Наименование товара (изделия), основные технические данные	Количество (шт.)	Единица измерения	Срок отгрузки
1	2	3	4	5
9(11)	А(20)	9(7)		

Рис. 6.3. Форма входного документа ПГП8

ление статистического отчета о выполнении плана поставок продукции», «Оперативно-календарное планирование отгрузки продукции».

Описание входной оперативной информации. Документ «План производства продукции» (рис. 6.1) составляется плановым отделом на квартал с разбивкой по месяцам. Он содержит шифр изделия, количество изделий на квартал, и в том числе по месяцам. Максимальное количество документострок не превосходит 200. «Разнарядка на поставку продукции» (рис. 6.2) поступает в отдел сбыта от фондодержателей каждый квартал в течение года. Указанный документ содержит шифр, наименование и адрес грузополучателя, шифр, наименование и адрес плательщика,

Таблица 6.1

Наименование реквизита	Обозначение реквизита	Длина в символах	Диапазон изменения	Единица измерения	Тип реквизита
1	2	3	4	5	6
Шифр изделия	ШИ	11	10000000—99999999999	—	Ц
Количество	К	7	10—9999999	шт.	Ц
Номер заказ-наряда	НОМЗ	13	1—999999999999	—	Ц
Наименование изделия	НИ	20	—	—	А
Шифр грузополучателя	ШГР	4	1—9999	—	Ц
Наименование и адрес грузополучателя	НГР	60	—	—	А
Процент торговой скидки	ТС	2,2	0—99,99	—	Ц
Шифр плательщика	ШПЛ	4	1—9999	—	Ц
Наименование и адрес плательщика	НПЛ	60	—		А
Местонахождение банка	БАНК	72	—		А
Расчетный счет	РС	9	1—999999999		Ц
Шифр фондодержателя	ШФОН	4	1—9999		Ц
Наименование фондодержателя	НФОН	40	—	—	А
Шифр внешнеторгового объединения	ВО	2	1—99		Ц
Вид продукции	ВП	2	1—99		Ц

местонахождение банка, номер расчетного счета, шифр изделия, количество изделий на квартал, и в том числе по месяцам, процент торговой скидки, шифр и наименование фондодержателя. Максимальное количество документострок — 50 тыс.

«Заказ-наряд на поставку продукции на экспорт» (рис. 6.3) поступает в отдел сбыта ежеквартально в течение года от внешнеторгового объединения. Он содержит шифр, наименование и количество изделий на квартал с разбивкой по месяцам, наименование страны-заказчика. Максимальное количество документострок — 30 тыс.

Для каждого реквизита указывается его формат и диапазон изменения значений (минимальное и максимальное значения). Непосредственно на формах входных документов приведены форматы всех обрабатываемых реквизитов. Например, обозначение вида 9(11) означает, что соответствующий реквизит является цифровым и занимает 11 символов, а обозначение А(20) указывает, что реквизит алфавитно-цифровой и состоит из 20 символов.

Описание реквизитов входной информации приведено в табл. 6.1. При этом для каждого реквизита приводится его условное обозначение, длина в символах, диапазон изменения (минимальное и максимальное значения) для цифровых реквизитов, единица измерения для количественных реквизитов и тип реквизита (Ц — цифровой, А — алфавитно-цифровой).

Шифр изделия представляет собой составной реквизит вида ВСД, где В — шифр механизма (четыре цифровых символа); С — вид покрытия (один символ); Д — вариант изделия (шесть символов).

Описание нормативно-справочной информации. «Справочник потребителей продукции» (рис. 6.4) составляется к началу планируемого года и включает шифр и наименование покупателя, шифр и наименование фондодержателя, номер договора и дату его заключения. Максимальное количество документострок — 2 тыс.

«Справочник цен по изделиям» (рис. 6.5) составляется плановым отделом и содержит следующие реквизиты: номер группы, шифр изделия, цены оптовые и розничные, плановую себестоимость по кварталам, наименование изделия, номер прейскуранта и дату его утверждения. Максимальное количество документострок — 200.

«Спецификация на поставку продукции» является приложением к договору на поставку и содержит следующие реквизиты: шифр покупателя, наименование покупателя, шифр изделия, количество изделий на год с разбивкой по кварталам. Этот справочник представлен в виде массива на магнитных лентах. Длина записи постоянна и равна 6 ячейкам. Ориентировочное число записей в массиве составляет 40 тыс.

Описание выходной информации. Выходной документ «План поставок продукции на год» (рис. 6.6) должен рассчитываться один раз в год и выдаваться в четырех экземплярах. По одному экземпляру поступает в планово-экономический и производственно-диспетчерский отделы завода, где они используются при раз-

Справочник потребителей продукции

Получатель		Номер договора и дата его заключения	Фондодержатель	
шифр	наименование		шифр	наименование
1	2	3	4	5
9(4)	A(30)	A(10)	9(4)	A(40)

Рис. 6.4. Форма нормативно-справочного документа НСИ1

Справочник цен по изделиям

Номер группы	Шифр изделия	Розничная цена (руб.)	Оптовая цена (руб.)	Плановая себестоимость по кварталам (руб.)				Наименование изделия	Номер прейскуранта и дата его утверждения
				I	II	III	IV		
1	2	3	4	5	6	7	8	9	10
9(1)	9(11)	9(3), 9(2)	9(2), 9(2)	9(2), 9(2)	9(2), 9(2)	9(2), 9(2)	9(2), 9(2)	A(20)	A(50)

Рис. 6.5. Форма нормативно-справочного документа НСИ2

План поставок продукции на _____ год

Шифр изделия	Годовая поставка		Квартальная поставка							
	всего	в том числе на внутренний рынок	всего на I кв.	в том числе на внутренний рынок	всего на II кв.	в том числе на внутренний рынок	всего на III кв.	в том числе на внутренний рынок	всего на IV кв.	в том числе на внутренний рынок
1	2	3	4	5	6	7	8	9	10	11
9(11)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)
Итого по виду покрытия	×	×	×	×	×	×	×	×	×	×
Итого по шифру механизма	×	×	×	×	×	×	×	×	×	×
Всего . . .	×	×	×	×	×	×	×	×	×	×
Стоимость в оптовых ценах (тыс. руб.)	×	×	×	×	×	×	×	×	×	×

Рис. 6.6. Форма выходного документа ПГП1

План поставок продукции по потребителям на _____ квартал (месяц) _____ года

Шифр				Количество на квартал	В том числе по месяцам		
покупателя	фондодержателя	грузополучателя	изделия		1	2	3
1	2	3	4	5	6	7	8
9(4)	9(4)	9(4)	9(11)	9(7)	9(7)	9(7)	9(7)
Итого по фондодержателю				×	×	×	×

Рис. 6.7. Форма выходного документа ПГП2

Соответствие плана производства портфелю заказов на_____квартал (месяц)_____года

Шифр изделия	План производства				Заказ по разнарядкам				Отклонение			
	количество на квартал	в том числе по месяцам			количество на квартал	в том числе по месяцам			количество на квартал	в том числе по месяцам		
		1	2	3		1	2	3		1	2	3
1	2	3	4	5	6	7	8	9	10	11	12	13
9(11)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)	9(7)

Рис. 6.8. Форма выходного документа ПГП3

Ведомость соответствия разнарядок спецификациям на_____число_____месяца

Шифр изделия	Подлежит поставке по спецификации	Заказ по разнарядкам	Отклонение
1	2	3	4
9(11)	9(7)	9(7)	9(7)
Итого по виду покрытия	×	×	×
Итого по шифру механизма	×	×	×
Всего . . .	×	×	×

Рис. 6.9. Форма выходного документа ПГП4

Ведомость отклонений разнарядок от спецификаций на_____число_____месяца

Шифр покупателя	Шифр изделия	Подлежит поставке по спецификациям	Заказ по разнарядкам	Отклонение
1	2	3	4	5
9(4)	9(11)	9(7)	9(7)	9(7)
Итого по виду покрытия		×	×	×
Итого по шифру механизма		×	×	×
Всего . . .		×	×	×

Рис. 6.10. Форма выходного документа ПГП5

работке плана производства, и по одному — в отдел сбыта и финансовый.

Он содержит сведения о годовом плане поставки готовой продукции с разбивкой по кварталам.

Порядок расположения строк данной формы следующий: сначала должны печататься шифры изделий одного механизма определенного вида покрытия с выводом итога по виду покрытия; затем следуют изделия этого же механизма, но другого покрытия с выводом соответствующего итога; в заключение выводятся итоговые данные по этому механизму; затем располагаются шифры изделий другого механизма, но вышеуказанный порядок расположения строк сохраняется.

В конце формы выводится итог по всем видам изделий и дается стоимостная оценка их в оптовых ценах. Итог подводится по реквизитам, отмеченным на форме документа символом ×. Максимальное количество строк — 1000.

Выходной документ «План поставок продукции по потребителям на квартал (месяц)» (рис. 6.7) печатается ежеквартально. В нем приводятся данные о плане поставок на квартал с разбивкой по месяцам в разрезе шифров покупателей, фондодержателей и грузополучателей. Подводятся итоги по шифрам фондодержателей. Максимальное количество строк — 2 тыс. Данный документ поступает в планово-экономический и производственно-диспетчерский отделы завода.

В выходном документе «Соответствие плана производства портфелю заказов на квартал (месяц)» (рис. 6.8) содержатся данные, показывающие отклонения плана производства от портфеля заказов на каждый квартал с разбивкой по месяцам. Этот документ выдается по запросу. Максимальное количество строк — 200.

Выходной документ «Ведомость соответствия разнарядок спецификациям» (рис. 6.9) формируется для контроля обеспеченности спецификаций разнарядками на каждый квартал года. Из данного документа можно получить сведения, по каким шифрам изделий спецификации не обеспечиваются разнарядками. Этой формой пользуется финансово-сбытовой отдел завода. Максимальное количество строк — 500.

Выходной документ «Ведомость отклонений разнарядок от спецификаций» (рис. 6.10) содержит отклонения объемов заказов по разнарядкам от запланированных в спецификациях. Максимальное количество строк — 2 тыс.

Описание накапливаемой и промежуточной информации. Накапливаемой информации для решения данной задачи нет. При решении задачи нужно выделить два массива промежуточной информации: массив плана поставок и массив месячных планов.

Запись массива плана поставок должна содержать следующие реквизиты:

Шифр изделия

План на I кв.
План на II кв.
План на III кв.
План на IV кв.
} Внутренний рынок

План на I кв.
План на II кв.
План на III кв.
План на IV кв.
} Экспорт

Ориентировочное количество записей в массиве 200.

Запись массива месячных планов должна содержать следующие реквизиты:

Шифр фондодержателя
Шифр плательщика
Шифр грузополучателя
Вид продукции
Шифр изделия
План на I месяц
План на II месяц
План на III месяц.

Ориентировочное количество записей в массиве 40 тыс.

Алгоритм решения задачи. На основании заказ-нарядов о поставке продукции на экспорт (ПГП8) и спецификации на поставку продукции на внутренний рынок (НСИ3) формируется годовой (с разбивкой по кварталам) план поставок продукции потребителям P_i:

$$p_i = \sum_{j} p_{ij},$$

где p_{ij} — заказ j-м потребителям i-го изделия.

Стоимость всех заказов в оптовых ценах (C) равна:

$$C = \sum_{L} \sum_{j} p_{ij} c_i,$$

где c_i — оптовая стоимость i-го изделия. Годовой план поставок продукции выдается на печать в виде документа ПГП1 с подведением следующих итогов: по виду покрытия, шифру механизма, всего, стоимость всех изделий.

Ежеквартально годовой план поставок уточняется разнарядками на поставку продукции (ПГП7), на основании которых формируется квартальный план поставок по каждому потребителю и выдается выходной документ ПГП2 с подведением итогов по шифрам грузополучателя и фондодержателя. На основании разнарядок на поставку продукции (ПГП7) устанавливается соответствие квартальных заказов годовому плану поставок.

Отклонения для каждого покупателя выдаются в виде документа ПГП5, а соответствия расшифровываются по шифрам изделия в документе ПГП4. На основании квартального плана производства изделий устанавливается обеспеченность квартального плана поставок, что находит отражение в выходном документе ПГП3.

6.3. ИНФОРМАЦИОННАЯ УВЯЗКА

Система обработки данных в АСУП предполагает решение ряда задач. Одной из существенных особенностей задач АСУП является их информационная преемственность, заключающаяся в том, что информация, полученная в одной задаче, должна использоваться в других. Объясняется эта особенность тем, что любая совокупность задач АСУП представляет некоторую модель информационно-экономической системы предприятия, которая в свою очередь является сложной взаимосвязанной системой. В этой связи при разработке системы обработки данных особую важность приобретает проблема информационной увязки задач. В этой проблеме, сложность которой усугубляется большим числом задач и обработкой большой номенклатуры реквизитов, следует выделить следующие аспекты.

Во-первых, необходимо, чтобы в каждой задаче (подсистеме) готовились промежуточные данные, используемые затем в других задачах и подсистемах, а в некоторых случаях — и для последующего решения данной задачи. Во-вторых, нужно обеспечить терминологическое единство показателей и реквизитов в рамках всей системы. В-третьих, следует стремиться к исключению дублирования и повторяемости показателей и реквизитов в различных массивах. Основная цель этого — добиться, чтобы всякий массив данных вводился, просматривался и обрабатывался минимальное количество раз. Отметим, что стремление к устранению дублирования не должно увеличивать затраты машинного времени на организацию массивов, их корректировку и решение задач. Поэтому в отдельных случаях может оказаться разумным для некоторой задачи или комплекса задач иметь массивы, данные которых содержатся и в других массивах.

Решение этих вопросов далеко не тривиально, тем более что они связаны с выбором структуры массивов, которая в свою очередь существенно зависит от конфигурации и мощности ЭВМ.

В настоящее время отсутствует четкая методика проведения информационной увязки задач. Обычно полагаются на интуицию специалистов, имеющих опыт обработки данных. Можно указать лишь некоторые рекомендации по выполнению этой работы.

Предварительное проведение информационной увязки осуществляется в процессе разработки постановок задач. На этом этапе проводится классификация информации на нормативно-

справочную, оперативную и выходную на уровне документов, т. е. указывается, к какому классу относятся соответствующие формы документов. При этом выбрасываются документы, содержащие одну и ту же информацию, но различающиеся по форме и наименованию реквизитов.

Для проведения унификации названий реквизитов используются словари экономических терминов. На этом же этапе указывается состав реквизитов промежуточной информации и наименование задач, в которых она образуется. В конечном итоге должны быть определены все информационные массивы, используемые или получаемые в подсистеме (задаче) и составлена схема информационной взаимосвязи задач (подсистем) на уровне информационных массивов.

В качестве примера приведем схему (рис. 6.11) информационной взаимосвязи задач по подсистеме управления сбытом и реализацией продукции, включающую следующие задачи:

— учет движения готовой продукции на складах (УГП);
— расчет плана поставок готовой продукции (ПГП);
— оперативный учет выполнения плана отгрузки (ОУП);
— учет отгруженной и реализованной продукции (УОР);
— расчет с финорганами по налогу с оборота (РФО);
— оперативно-календарное планирование отгрузки (ОКП);
— прогнозирование реализации и прибыли (ПРП);
— оперативный учет выполнения плана реализации и прибыли (ОУР);
— составление статистической отчетности о выполнении плана поставок (СТО).

На схеме в шестигранниках указываются наименования массивов нормативно-справочной информации, в пятигранниках — оперативной, в овалах — промежуточной, в прямоугольниках — наименования задач. В остальных блоках перечисляются шифры получаемых выходных документов. Стрелками показано, какие данные используются и какие промежуточные массивы и выходные документы получаются в процессе решения задач. На входе задачи УГП допускается многовариантность форм входной документации. Однако на основании этих форм в результате компоновки должен получиться массив оперативных данных о движении готовой продукции, состав реквизитов которого строго определен. С другой стороны, расчет задачи ПГП осуществляется на основании массивов оперативной информации заказов-нарядов, разнарядок, плана производства. При расчете этих задач используются массивы постоянной информации — справочники цен и спецификаций.

Результатом решения задачи ПГП является получение выходных документов:

— план поставок продукции на год (ПГП1);
— план поставок продукции на квартал (ПГП2);
— соответствие плана производства портфелю заказов (ПГП3);

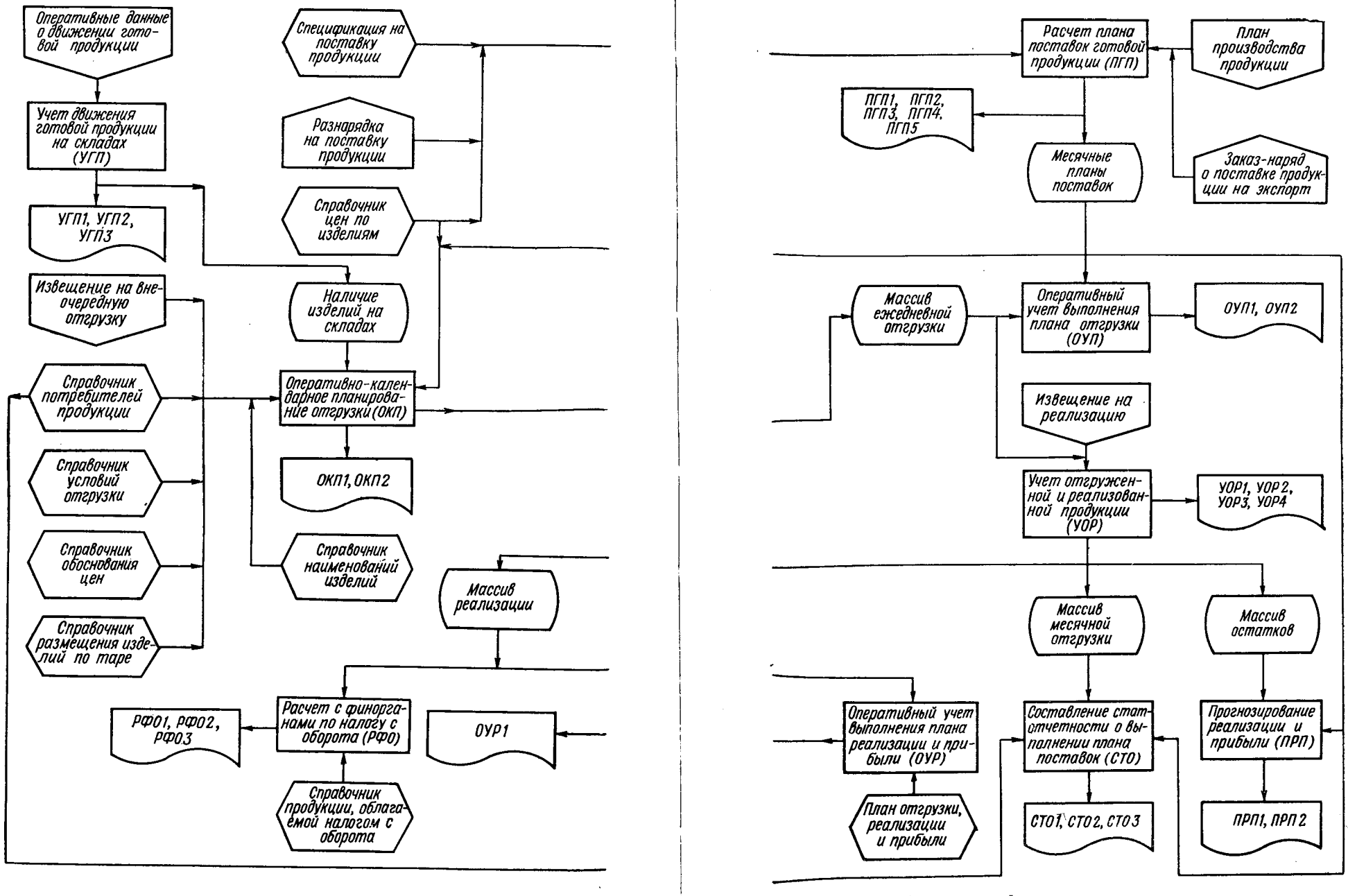


Рис. 6.11. Схема информационной взаимосвязи задач

— ведомость соответствия (несоответствия) разнарядок спецификациям (ПГП4);
— ведомость отклонения разнарядок от спецификаций (ПГП5).

В результате расчета УГП выдаются следующие выходные документы:
— ведомость прихода изделий на склад готовой продукции (УГП1);
— ведомость расхода изделий по складу (УГП2);
— ведомость движения изделий по складу готовой продукции (УГП3).

В результате расчета этих двух задач (УГП и ПГП), являющихся своеобразным входом в подсистему, параллельно с выдачей выходных документов формируются массивы наличия изделий на складах и месячных планов поставок, позволяющие практически без дополнительной оперативной информации осуществить расчет всего комплекса задач подсистемы, центральной из которых является ОКП.

Решение задачи ОКП позволяет осуществить на основании вышеуказанных промежуточных массивов и массивов нормативно-справочной информации (справочника потребителей продукции, справочника условий отгрузки, справочника размещения изделий по таре, справочника цен по изделиям и справочника наименований изделий) распределение имеющейся на складе продукции потребителям по заказанному ассортименту, распределение отгружаемой продукции по таре и выдачу документов на отгрузку продукции. Такими документами являются:
— платежное требование на отгрузку продукции (ОКП1);
— ведомость (реестр) отгружаемой продукции (ОКП2).

Полученный на магнитных лентах массив ежедневной отгрузки служит основой расчета всех последующих задач.

На основании данных задачи оперативного учета выполнения плана отгрузки (ОУП) осуществляется контроль за своевременной и ритмичной отгрузкой готовой продукции.

В качестве исходных данных используются промежуточные массивы, содержащие информацию о месячных планах поставок и ежедневной отгрузке, полученные в результате расчета задач ПГП и ОКП.

Результатом решения задачи ОУП является получение данных о выполнении плана отгрузки продукции по номенклатуре по внутреннему рынку (ОУП1) и экспорту (ОУП2), в которых содержится информация о плане отгрузки на месяц, ежедневной отгрузке за сутки и фактической отгрузке с начала месяца с указанием процента выполнения месячного плана отгрузки.

Целью задачи УОР является контроль за отгрузкой, реализацией и за состоянием неоплаченных требований на отгруженную продукцию.

В качестве исходных данных используются промежуточный массив ежедневной отгрузки, полученный в результате решения

задачи ОКП, и оперативная информация в виде извещения на реализацию.

При решении задачи образуются и накапливаются массивы, содержащие данные о реализованной, отгруженной и остатках отгруженной, но не реализованной продукции, которые используются при решении задач ОУР и ПРП. При этом получаются следующие выходные документы:

— свод отгруженной продукции по номенклатуре (УОР1);
— учет реализованной продукции (УОР2);
— учет отгруженной продукции (УОР3);
— остатки отгруженной продукции (УОР4).

В качестве исходной информации для решения задачи ОУР используются массив реализации и массив нормативно-справочной информации о плане отгрузки, реализации и прибыли.

Результатом решения задачи является получение выходного документа — выполнение плана реализации и прибыли (ОУР1), который содержит учетные данные о выполнении плана отгрузки, реализации и прибыли, позволяющие оперативно контролировать ход выполнения этого плана.

При решении задачи ПРП в качестве исходной информации используются массив, содержащий данные об остатках отгруженной, но не реализованной продукции, и массив месячных планов поставок, сформированные в задачах ОУР и ПГП.

Результатом решения задачи ПРП являются следующие выходные документы:

— ожидаемая реализация и прибыль (ПРП1);
— прогнозирование суммы реализации и прибыли (ПРП2).

В задаче РФО определяются суммы налога с оборота реализованной продукции.

В качестве исходной информации для решения данной задачи используются массив реализации, накапливаемый в результате решения задачи УОР, и справочник продукции, облагаемой налогом с оборота.

В результате решения задачи выдаются следующие выходные документы:

— расчет налога с оборота реализованной продукции (РФО1);
— свод продукции, облагаемой налогом с оборота (РФО2);
— свод продукции, не облагаемой налогом с оборота (РФО3).

Решение задачи СТО предусматривает учет выполнения планов поставок продукции по фондодержателям по внутреннему рынку и экспорту.

В качестве исходной информации используются массив месячной отгрузки из задачи УОР, массив месячных планов поставок из задачи ПГП и справочник потребителей продукции.

Результатом решения задачи являются следующие выходные документы:

— выполнение плана поставок продукции (СТО1);
— выполнение плана поставок продукции на экспорт (СТО2);

— недовыполнение плана поставок продукции на экспорт (СТО3).

Модульное (блочное) построение задач подсистемы обеспечивает возможность локальной реализации любой задачи данного комплекса.

6.4. СОСТАВЛЕНИЕ ПРИНЦИПИАЛЬНОЙ БЛОК-СХЕМЫ АЛГОРИТМА ЗАДАЧИ

Одним из наиболее ответственных этапов при разработке алгоритмов и программ задач АСУ является составление принципиальных блок-схем.

Блок-схема дает наглядное представление об алгоритме решения задачи, отражая технологический процесс обработки информации на ЭВМ.

В общем случае решение типичной задачи АСУП сводится к реализации следующих процедур обработки данных:

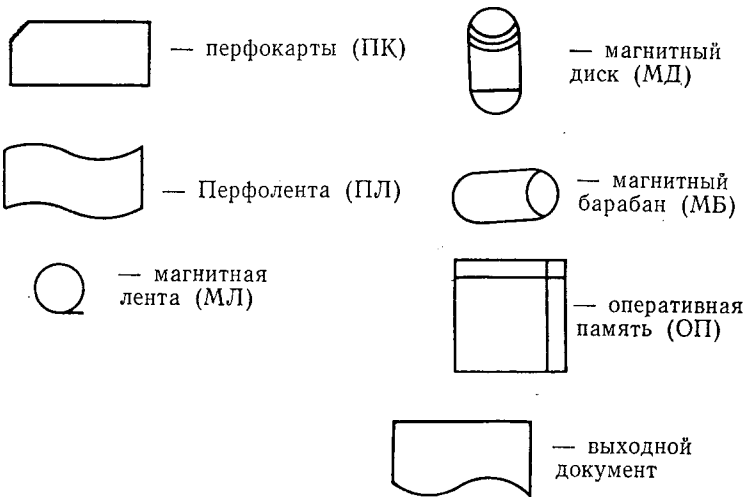
— ввод, контроль и компоновка массивов данных с перфоносителей;
— сортировка массивов информации в оперативной памяти и на магнитных лентах;
— корректировка массивов информации на магнитных лентах;
— преобразование массивов по заданному алгоритму;
— подготовка и выдача отчетов на устройство печати.

Способы реализации названных процедур могут быть различными. Во-первых, эти процедуры в каждой конкретной задаче могут программироваться заново, например, на языке символического кодирования или других средствах программирования. Во-вторых, эти процедуры могут быть реализованы в виде универсальных программ интерпретирующего типа [34]. В этом случае для выполнения соответствующей процедуры достаточно задать описание входных и выходных параметров. Универсальная программа сама настроится на значения этих параметров и произведет соответствующую обработку данных. В-третьих, эти процедуры могут быть реализованы в виде генераторов программ обработки данных. Если в предыдущем случае программа обработки выполнялась в режиме интерпретации, то при использовании генераторов в зависимости от значения параметров, характеризующих входные и выходные массивы, и свободных ресурсов ЭВМ генерируется программа, ориентированная на конкретную обработку. Следует отметить, что при использовании любых средств программирования обязательным этапом является разработка принципиальной блок-схемы.

При разработке схемы решения задачи каждая процедура изображается отдельным блоком. На каждом блоке может быть проставлен цифровой номер или идентификатор блока, обеспечивающий простоту ссылки на соответствующий блок. Слева от графического изображения процедуры приводятся изображения

машинных носителей для входных массивов с указанием их имен, справа — для выходных. Левее и соответственно правее изображения машинных носителей входных и выходных массивов могут располагаться комментарии. Параметры процедуры (их рекомендуется записывать в блоке комментариев справа) записываются первыми и отделяются горизонтальной чертой от прочего текста. Комментарии составляются в произвольной форме, однако они должны быть достаточными для детального понимания предусмотренной соответствующим блоком обработки информации.

Для изображения носителей входных и выходных данных будем использовать следующие условные графические обозначения [29]:

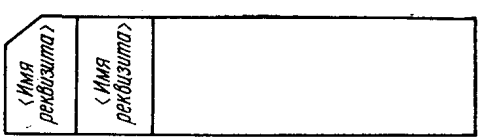


Внутри каждого из блоков может быть указано имя соответствующего массива или документа.

Структура записей массивов должна приводиться в комментариях, либо там должны быть соответствующие ссылки.

Для массива данных на ПК структура макетов, входящих в массив документов, изображается в виде:

МАКЕТ <шифр макета> ИЗ МАССИВА <имя массива>



В описании каждого реквизита должен указываться номер колонки начала реквизита на ПК.

Для массивов данных на ПЛ структура входящих в массив документов изображается в виде дерева подчиненности, узлами которого являются связки (см. разд. 3.1).

В общем случае каждый узел дерева подчиненности (связка) описывается в виде:

<имя связки> (<имя реквизита>,
<имя реквизита>, ...)

Для массива с разнотипными документами первой связкой обязательно должна быть связка нулевого уровня с именем ОЛ (операционный лист), а первым реквизитом — шифр документа (ШД).

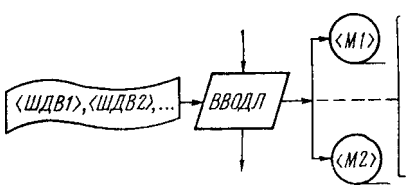
Для каждой задачи целесообразно составить перечень имен массивов, записей и реквизитов с указанием их полного названия, а для реквизитов — описание их форматов.

Рассмотрим графическое изображение каждой процедуры обработки данных.

Ввод, контроль и компоновка экономической информации с перфолент. Эта процедура может быть реализована с помощью программ, описанных в разделе 3.2.

Исходный массив на перфоленте может содержать документы разных типов с иерархической структурой. После ввода и соответствующего контроля (на корректность представления цифровой информации, диапазон изменения реквизитов, соответствие описанной иерархической структуры) документы компонуются в несколько массивов и записываются на МЛ.

Графическое изображение процедуры представляется в виде:



Компоновка:
ШДВ1 — (ИЗЫ1, ИЗЫ2)
ШДВ2 — (ИЗЫ3)...
Структура записей:
(<имя записи> — ИР1,
ИР2,...)...
НБП1 — КОНТР
НБП2 — ВЫВД

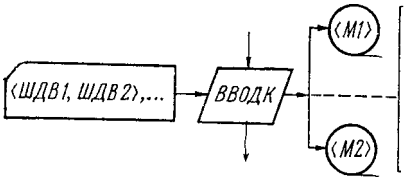
Здесь

М1, М2, ... — имена выходных массивов;
ШДВ1, ШДВ2, ...— шифры входных документов;
ИЗЫ1, ИЗЫ2, ... — имена выходных записей;
ИР1, ИР2, ... — имена реквизитов;
КОНТР и ВЫВД — имена нестандартных блоков пользователя (НБП).

Имена реквизитов компонуемых записей должы совпадать с именами соответствующих реквизитов во входных документах.

Ввод, контроль и компоновка экономической информации с перфокарт. Эта процедура может быть реализована с помощью программы, описанной в разделе 3.3. Массив на перфокартах может состоять из однотипных или разнотипных документов (макетов). В случае разнотипных макетов на ПК указывается шифр макета, при этом он должен занимать одно и то же поле на ПК для всех макетов. В процессе ввода зачастую проводится контроль цифровой информации на корректность представления и

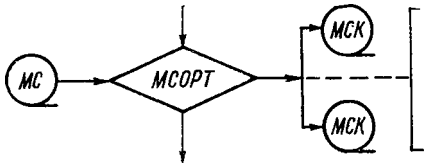
контролируется правильность подготовки информации (заполнение документов и перфорации) путем сравнения отперфорированных сумм со значением итогов, получаемых путем суммирования реквизитов. Проконтролированная информация может компоноваться в несколько массивов и записываться на магнитные ленты. Графическое изображение этой процедуры имеет вид:



Компоновка:
ШДВ1 — (ИЗЫ1, ИЗЫ2)
ШДВ2 — (ИЗЫ3)...
Структура записей:
(<имя записи> — ИР1, ИР2, ...)...
НБП1 — КОНТР
НБП2 — ВЫВД

Сокращения, применяемые на обозначении этой процедуры, совпадают с обозначениями аналогичной процедуры ввода с перфоленты. Имена реквизитов в компонуемых записях должны совпадать с именами соответствующих реквизитов во входных документах.

Сортировка массивов информации на магнитных лентах. Эта процедура может быть реализована с помощью программы, описанной в разделе 3.4. Она производит упорядочение записей в массиве по одному или нескольким ключевым реквизитам. Упорядочение записей осуществляется в возрастающей или убывающей последовательности с учетом знаков ключевых реквизитов или по их абсолютной величине. На выходе имеется возможность получить копию упорядоченного массива на другой магнитной ленте. Графическое изображение процедуры представляется в виде:

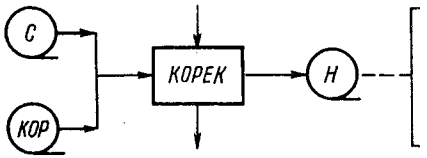


Сортировать:
ИР1(В), ИР2(В), ИР3(У) ...

Здесь МС — имя сортируемого массива;
ИР1, ИР2, ... — имена ключевых реквизитов;
В/У — признак сортировки (возрастание/убывание);
МСК — имя рассортированного массива.

Корректировка информации на магнитных лентах. Эта процедура может быть реализована с помощью программы, описанной в разделе 3.6. Предусмотренные в процедуре режимы позволяют выполнять над массивами такие операции, как вставку, удаление, замену, подсуммирование записей, слияние массивов, подчистку, проверку на корректность сортировки, получение копии массива. В процедуре могут участвовать как один, так и два

входных массива. Они могут иметь одинаковую или различную структуру записей. Графическое изображение процедуры имеет вид:



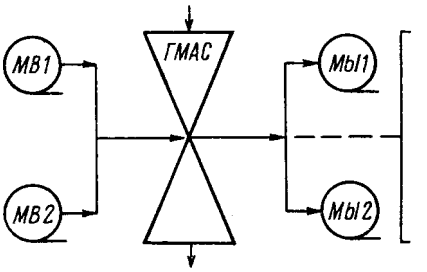
Здесь С — имя корректируемого массива (старого справочника);
КОР — имя массива корректур;
Н — имя выходного массива (нового справочника);
ИКР1, ИКР2, ... — имена сравниваемых (ключевых) реквизитов с указанием порядка рассортированности (В/У — возрастание/убывание);
ИР1, ИР2, ... — имена реквизитов, участвующих в подсуммировании (только для режимов СУМ[К]);
БЛ1, БЛ2, ... — имена нестандартных блоков пользователя (НБП).

Режим может быть одним из следующих: ВУЗ[К[Г]], СУМ[К], СЛН[К] (подробнее см. разд. 3.6).

Для режимов ВУЗ, СУМ, СЛН массив корректур отсутствует. Для режимов ВУЗК[Г] в комментариях описывается признак удаления или делается ссылка на его описание.

Преобразование массивов по заданному алгоритму. Данная процедура осуществляет обработку *n* входных массивов с получением на выходе *m* выходных массивов. В процессе обработки выполняются необходимые операции над данными исходных массивов и формируются выходные массивы.

Процедура графически изображается следующим образом:



Здесь МВ1, МВ2, ... — имена входных массивов;
МЫ1, МЫ2, ... — имена выходных массивов;
И1, И2, ... — имена реквизитов выходных записей;
ИР1, ИР2, ...— имена сравниваемых реквизитов.

Вид обработки может быть одним из следующих:

П — последовательная, когда записи в массиве обрабатываются одна за другой;

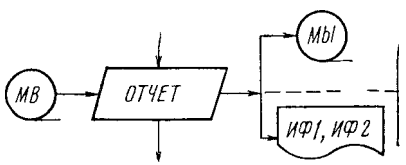
Г — групповая. При этом виде обработки массив разбивается на некоторые подмножества в зависимости от значения ключевых реквизитов. Для каждого из подмножеств (групп)записей возможно многократное обращение для использования в обработке;

ПН — последовательная с накоплением. Накопление реквизитов рядом стоящих записей осуществляется в зависимости от значения ключевых реквизитов;

ГН — групповая с накоплением.

Чтобы не усложнять блок-схему, алгоритм преобразования целесообразно описать отдельно.

Составление отчетов и выдача на устройство широкой печати. Эта процедура может быть реализована с помощью программ, описанных в разделе 3.7. Исходными данными для процедуры являются массивы данных, состоящие из однотипных или разнотипных записей, расположенных на магнитной ленте. В процессе составления отчета проводится подведение итогов по изменению некоторых ключевых реквизитов, по странице и в конце отчета. Составленный отчет выдается на печать в удобочитаемом виде выходного документа с оформлением титульных листов, верхних и нижних шапок. Процедура графически изображается так:

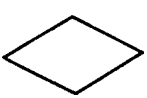


Сравнение для итогов: (ИЗ1—ИР1, ИР2, ...)...

Здесь МВ — имя входного массива;
МЫ — имя выходного массива;
ИЗ1, ИЗ2, ...— имена входных записей;
ИР1, ИР2, ... — имена сравниваемых реквизитов;
ИФ1, ИФ2, ... — имена выходных документов (формы выходных документов приводятся на специальных бланках).

В комментариях могут приводиться ссылки на описание алгоритма составления отчета.

При составлении блок-схемы, кроме указанных блоков, используются следующие графические изображения:



— разветвление вычислительного процесса.

— начало-конец этапов обработки информации.

В логических блоках целесообразно показывать, по каким условиям идет разветвление алгоритма решения задачи. В некоторых случаях можно прибегнуть к записи логической формулы.

Ссылки на элементы данных, приведенных на блок-схеме, осуществляются по их именам. Формулы получения вычисляемых реквизитов записываются в обычном виде, например,

$$КД = РМ \times К + Д,$$

где КД, РМ, К, Д — имена реквизитов.

Для определения соотношения между реквизитами используются операции отношения $>$, $<$, $=$, $\neq$, $\geqslant$, $\leqslant$, а также логические операции $\wedge$, $\vee$, $\urcorner$.

Соотношение между записями определяется отношением между сравниваемыми реквизитами этих записей в порядке убывания их старшинства с учетом порядка рассортированности. Отношение между массивами в процессе обработки определяется отношением записей этих массивов.

Для составления принципиальной блок-схемы необходимо:

— тщательно изучить постановку задачи и ее алгоритм;
— выписать все входные и выходные документы, выбрать из них необходимые реквизиты;
— составить первый вариант блок-схемы;
— проанализировать блок-схему с целью минимизации времени решения задачи, рассмотреть формы записей с целью сокращения избыточной информации;
— составить окончательный вариант алгоритма в соответствии с блок-схемой;
— распределить внешние запоминающие устройства (МЛ), стремясь по возможности использовать минимальное число МЛ, а также сократить их замены во время решения задачи.

К составлению рабочих блок-схем можно приступать только после составления принципиальных. Принципиальная блок-схема призвана помочь программисту и постановщику глубже изучить задачу, быстрее выявить недостатки ее постановки и устранить их на ранней стадии.

Отметим, что принципиальная блок-схема существенно помогает быстрее вникнуть в сущность задачи и освоить реализацию машинного алгоритма на ЭВМ. Блок-схема дает возможность судить о таких параметрах задачи, как ее объем (количество программ), количество входной и выходной информации и др., тем самым позволяя определить специфические особенности данной задачи по сравнению с другими задачами такого же типа. Блок-схема существенно упрощает корректировку программ, если в ходе эксплуатации в ней возникает необходимость.

Кроме того, блок-схема позволяет определить трудоемкость работ по программированию, равномерно загрузить программистов с учетом их подготовленности, дает возможность одновременно программировать все части задачи, связь между которыми ясно видна из блок-схемы.

Подключение форм машинных документов и записей к функциональным блокам дает возможность наглядно проследить преобразование массивов, логику алгоритма по всей задаче, увязать их между всеми частями задачи, избежать дублирования массивов, свести их число в решаемой задаче к минимальному, избежать повторения отдельных частей алгоритма и приблизиться к оптимальному варианту алгоритма решения задачи.

Известно, что значительное время теряется при внесении изменений в алгоритм счета при привязке задачи к конкретному объекту (имеется в виду типовая задача). Общая блок-схема решения задачи помогает значительно сократить время привязки задачи, так как позволяет точно определить, куда в алгоритме можно внести изменения с наибольшей эффективностью, и установить последовательность их внесения. Блок-схема позволяет также установить полноту вносимых изменений.

Наличие таких блок-схем по разрабатываемым задачам явится научным багажом для дальнейших обобщений, выделения и популяризации наиболее прогрессивных и рациональных приемов и методов реализации алгоритмов задач АСУП.

Рассмотрим блок-схему (рис. 6.12) и машинный алгоритм задачи «Расчет плана поставок готовой продукции» (ПГП). Используемые в блок-схеме имена документов, массивов реквизитов описаны в разделе 6.2. Кроме этого, в комментариях использованы следующие обозначения реквизитов:

ПЛАН(i) — план поставок на i-й квартал;

ПЛАНВ (i) — план поставок на i-й квартал по внутреннему рынку;

ПЛ(i) — план производства на i-й квартал.

На первом этапе вычислительного процесса решения задачи ПГП осуществляется компоновка входных документов заказа-наряда (форма ПГП8) в массив ЗНАР (блок *1*), который на следующем шаге (блок *2*) сортируется в порядке возрастания ШИ. В результате получается массив ЗНАРК.

На основании массивов ЗНАРК и массивов спецификаций (НСИ3) и справочника цен (НСИ1) процедурой преобразования массивов по заданному алгоритму (блок *3*) осуществляется формирование плана поставок продукции (массив ПЛАН). Сформированный массив ПЛАН в блоке *4* сортируется по возрастанию значений составного реквизита С, В, Д. На основании упорядоченного массива ПЛАНК выдается выходной документ ПГП1 (блок *5*).

На следующем этапе (блок *6*) производится компоновка данных разнарядок на поставку продукции (форма ПГП7) в массив РАЗН, который затем упорядочивается по возрастанию значений реквизитов ШФОН, ШГР, ШИ (блок *7*). Полученный массив РАЗНК корректируется (блок *8*) по мере поступления новых документов ПГП7, в результате получается массив ПЛАК, на основании которого выдается на печать документ ПГП2

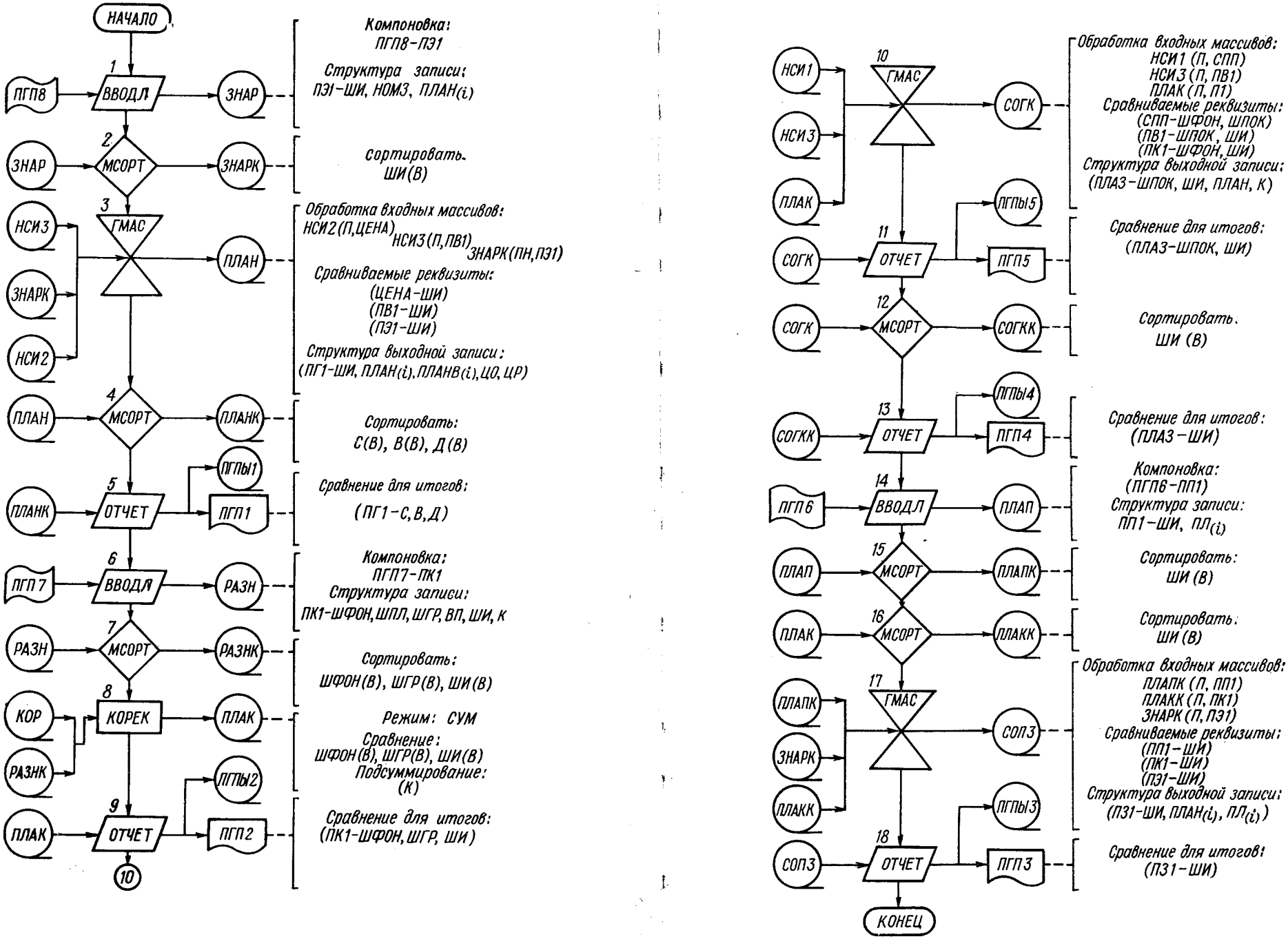


Рис. 6.12. Принципиальная блок-схема задачи ПГП

На основании массива спецификаций (НСИ3), справочника потребителей (НСИ1) и массива (ПЛАК) формируется (блок *10*) массив обеспеченности спецификаций разнарядками (СОГК). На основании этого массива выдается (блок *11*) выходной документ ПГП5. Массив СОГК упорядочивается по возрастанию ШИ (блок *12*). Рассортированный массив СОГКК предназначен для печати документа ПГП4 (блок *13*). В блоке *14* производится компоновка данных плана производства (ПГП6) в массив ПЛАП. Далее массивы ПЛАКК и ПЛАПК упорядочиваются (блоки *15, 16*) в порядке возрастания ШИ. На основании упорядоченных по ШИ массивов ПЛАПК и ПЛАКК и ЗНАРК определяется (блок *17*) обеспеченность квартальных заказов на поставку продукции. Полученный массив СОПЗ предназначен для выдачи выходного документа ПГП3 (блок *18*).

6.5. СОСТАВЛЕНИЕ РАБОЧИХ БЛОК-СХЕМ И ПРОГРАММ

При разработке рабочей блок-схемы программы основное внимание обращается на составление подробного формального описания процесса обработки данных с учетом специфики ее реализации на конкретной вычислительной машине. Поэтому здесь используют несколько другие символы, чем те, которые применяются при составлении принципиальной блок-схемы. Например, прямоугольники употребляются для обозначения операций обработки данных (арифметических, операций пересылки и др.). Параллелограммы — для обращения к внешним устройствам, включая операции вызова (занесения) логических записей.

Операции условного периода обозначаются в виде ромбов с указанием логического отношения (условия), по которому осуществляется переход к выполнению той или иной операции. Стрелками показывается последовательность их выполнения. Для удобства ссылок каждому блоку присваивается номер.

На рис. 6.13 приведена рабочая блок-схема, которая соответствует блоку *17* принципиальной блок-схемы задачи ПГП. В блоке *1* осуществляется закрепление магнитных лент за входными и выходными массивами, а выдача сообщений на пишущую машинку пульта оператора о произведенном закреплении — в блоке *2*. В блоке *3* открываются входные и выходной массивы ПЛАПК, ЗНАРК, ПЛАКК, СОПЗ. Необходимые для работы стандартные программы загружаются в блоке *4* с ленты системы в оперативную память. В блоках *5÷11* производится последовательное чтение записей массивов ЗНАРК, ПЛАПК, ПЛАКК в рабочее поле с проверкой на конец соответствующего массива. Если какой-нибудь из массивов окончен, то в рабочее поле для записи данного массива пересылается константа 9...9. В блоке *24* осуществляется сравнение шифров изделий в записях массивов ПЛАКК и ЗНАРК. Если шифр изделия записи ПЛАКК, находя-

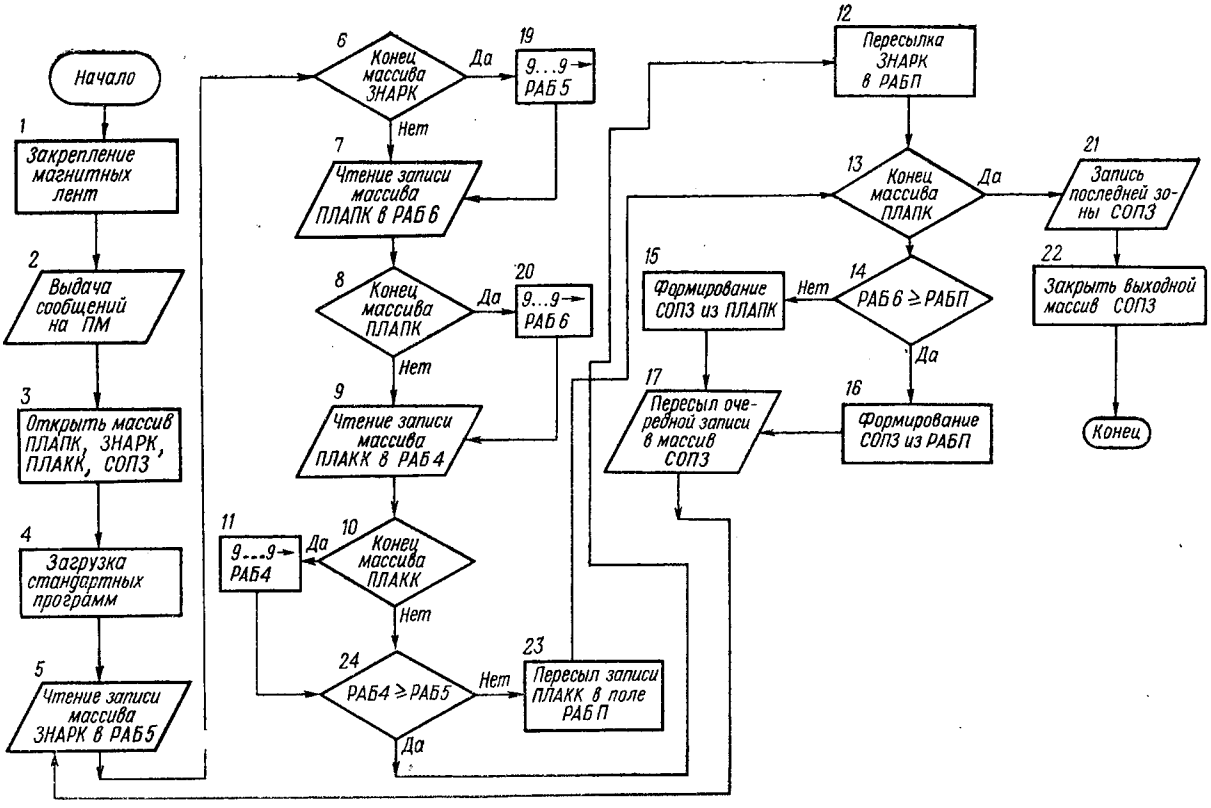


Рис. 6.13. Рабочая блок-схема программы ПГПП7

щейся в рабочем поле РАБ4, больше или равен шифру изделия записи ЗНАРК из рабочего поля РАБ5, то в блоке *12* осуществляется пересылка записи ЗНАРК в рабочее поле РАБП. В противном случае пересылается в блоке *23* запись из массива ПЛАКК в поле РАБП. В блоке *13* осуществляется проверка на конец массива ПЛАПК. Если массив ПЛАПК не окончен, то в блоке *14* сравниваются шифры изделий очередной записи из массива ПЛАПК, размещенной в поле РАБ6, и записи, находящейся в поле РАБП. В случае, если шифр изделия записи ПЛАПК больше или равен шифру изделия записи из рабочего поля РАБП, в блоке *16* формируется запись выходного массива СОПЗ на основании последней, иначе в блоке *15* производится формирование записи массива СОПЗ из записи массива ПЛАПК. В блоке *17* сформированная запись заносится в массив СОПЗ и управление передается на чтение очередной записи массива ЗНАРК (блоку *5*). Управление по концу массива ПЛАПК передается блоку *21*, где производится запись последней зоны массива СОПЗ, а в блоке *22* выходной массив СОПЗ закрывается.

На основании блок-схемы составляется рабочая программа, приведенная ниже. Каждый блок из блок-схемы 6.13 реализуется в рабочей программе с помощью группы команд. В тексте программы начало такой группы команд отмечено номером блока в соответствии с рабочей блок-схемой. Например, группе команд закрепления МЛ с этикеткой З соответствует блок *1*.

Одновременно с написанием программ должна осуществляться подготовка к их отладке. Отладка программ производится с помощью контрольного примера. Методы отладки зависят от используемых при программировании алгоритмических языков. При программировании на ССК рекомендуется каждый отдельный блок, интересующие программиста отдельные величины, массивы и т. д. именовать этикетками с целью упрощения отладки.

Для отладки программ используются отладочные программы. Например, для ЭВМ «Минск-32» используются отладочные программы ОП1 и ОП2. Эти программы позволяют вывести промежуточные результаты, полученные после выполнения той или другой команды, заменить (вставить) в процессе выполнения в программу некоторые команды, выполнить отдельные участки программы и т. п.

При решении задачи процесс управления переходом от выполнения одного задания к другому и от одного шага выполняемого задания к другому осуществляется управляющей программой. При окончании выполнения задания управление получает программа управления заданиями, которая проверяет карту, идентифицирующую новое задание, и передает управление новому заданию. Такая организация работы исключает необходимость вмешательства оператора в процессе смены заданий, что является весьма существенным для рациональной организации вычислительного процесса.

Программа проверки обеспеченности квартальных заказов ПГП17

Этикетка	КОП	Адреса и замечания
	ЗАГЛ	ПРОВЕРКА ОБЕСПЕЧЕННОСТИ КВАРТАЛЬНЫХ ЗАКАЗОВ
	БАЗ	0
НАЧ	РЗВ	3
ПАР	НОП	
	РИП	16
	СУ	4; +16
З	ЗАКР	МЛ, 1 блок 1
	ЗАКР	МЛ, 2
	ЗАКР	МЛ, 3
	ЗАКР	МЛ, 4
КВАРТ	ЖОО	МЛ, 1 блок 2
	КОСЛ5	0; ТЕКС1; 6
	ЖОО	МЛ, 2
	КОСЛ5	0; ТЕКС2; 6
	ЖОО	МЛ, 3
	КОСЛ5	0; ТЕКС3; 6
	ЖОО	МЛ, 4
	КОСЛ5	0; ТЕКС4; 6
	ЗГР	ЛС
	КА	0; ОТВМЛ
ПОДВ	ПН	МЛ, 1 блок 3
	ПН	МЛ, 2
	ПН	МЛ, 3
	ПНЖ	МЛ, 4
	ИП	ОТВМЛ; 1
	КА	Д; А
	ИП	ОТВМЛ; 1
	КА	Д1; А1
	ИП	ОТВМЛ; 1
	КА	Д2; А2
	ЗГР	ЛС
	КА	0; ОТЫМЛ
	ИП	ОТЫМЛ; 1
	КА	Д3; А3
	ЗГР	ЛС блок 4

Этикетка	КОП	Адреса и замечания
	КА	0; ВМЛ
	ЗГР	ЛС
	КА	0; ЫМЛ
	ЗГР	ЛС
	КА	0; ЧТЗ
	ЗГР	ЛС
	КА	0; ЗПЗ
	ЗГР	ЛС
	КА	О; ЗАЫ
	П	М; ПР1
	П	М; ПР2
	П	М; ПР3
ЧИТ	ПС	+ 0; ПР2 блоки 5, 6
	ИРН	* + 4
	ИП	ЧТЗ; 2
	КА	Д1; А1
	КА	0; ВМЛ
Ч1	ПС	+ 0; ПР3 блоки 7, 8
	ИРН	* + 4
	ИП	ЧТЗ; 2
	КА	Д2; А2
	КА	0; ВМЛ
Ч2	ПС	+ 0; ПР1 блоки 9, 10
	ИРН	* + 4
	ИП	ЧТЗ; 2
	КА	Д; А
	КА	0; ВМЛ
	ВФ	РАБ5 + 4; РАБ4 + 5 блок 11
Ч3	ИНРН	ПЕР
	ПС	РАБ4 + 5; КОНСТ
	ИРН	БЛ1
	ВФ	РАБ5 + 5; РАБ4 + 6
	ИНРН	ПЕР
	ВФ	РАБ5 + 3; РАБ4 + 4
ПЕР	ИЗН	БЛ1; * + 1

Этикетка	КОП	Адреса и замечания	
	ГРУП	+ 7	блок 23
	П	РАБ4 + 4; РАБП	
	П	М; ПР1	
	П	+ 0; ПР2	
	ИН	БЛ2; БЛ2	
БЛ1	ГРУП	+ 7	блок 12
	П	РАБ5 + 3; РАБП	
	П	М; ПР2	
	П	+ 0; ПР1	
БЛ2	ВФ	РАБП + 1; РАБ6 + 4	блок 13
	ИНРН	ПЕР1	
	ПС	РАБ6 + 4; КОНСТ	
	ИРН	КОНЕЦ	
	ВФ	РАБП + 2; РАБ6 + 5	блок 14
	ИНРН	ПЕР1	
	ВФ	РАБП; РАБ6 + 3	
ПЕР1	ИЗН	БЛ3; * + 1	
	П	+ 0; ПР1	блок 15
	П	+ 0; ПР2	
	П	М; ПР3	
	ГРУП	+ 6	
	П	РАБ6 + 5; РАБ8 + 3	
	П	+ 0; РАБ8 + 10	
	П	+ 0; РАБ8 + 11	
	П	+ 0; РАБ8 + 12	
	П	+ 0; РАБ8 + 13	
	ИН	БЛ4; БЛ4	
БЛ3	П	+ 0; ПР3	блок 16
	П	РАБП; РАБ8 + 3	
	П	РАБП + 1; РАБ8 + 4	
	П	РАБП + 2; РАБ8 + 5	
	П	+ 0; РАБ8 + 6	
	П	+ 0; РАБ8 + 7	
	П	+ 0; РАБ8 + 8	
	П	+ 0; РАБ8 + 9	

Этикетка	КОП	Адреса и замечания	
	ГРУП	+ 3	
	П	РАБП + 3; РАБ8 + 10	
БЛ4	П	ДЛ; РАБ8 + 1	блок 17
	ИП	ЗПЗ; 2	
	КА	Д3; А3	
	КА	0; ЫМЛ	
	ИН	ЧИТ; ЧИТ	
КОН	П	КОНСТ; РАБ4 + 5	блок 18
	ИЧ	Ч3; + 0	
КОН1	П	КОНСТ; РАБ5 + 4	блок 19
	ИЧ	Ч1; + 0	
КОН2	П	КОНСТ; РАБ6 + 4	блок 20
	ИЧ	Ч2; + 0	
КОНЕЦ	ИП	ЫМЛ; 1	блок 21
	КА	Д3; А3	
	ИПИ	ЗАЫ; 1	блок 22
	КА	Д3; А3	
	ВУ	4; + 16	
	ОСВ	МЛ, 1	
	ОСВ	МЛ, 2	
	ОСВ	МЛ, 3	
	ОСВ	МЛ, 4	
	ВЫХ	НАЧ; 0	конец
ТЕКС1	КТ	УСТАН	
		. МЛ	
		С МАС	
		СИВОМ	
		ПЛАК	
		К.	
ТЕКС2	КТ	УСТАН	
		. МЛ	
		С МАС	
		СИВОМ	
		ЗНАР	
		К.	

Этикетка	КОП	Адреса и замечания
ТЕКС3	КТ	УСТАН
		. МЛ
		С МАС
		СИВОМ
		ПЛАП
		К.
ТЕКС4	КТ	УСТАН
		. МЛ
		ДЛЯ М
		АССИВ
		А СОП
		З.
ОТВМЛ	ОДРЗ	ОТВМЛ
	НОП	
	НОП	
Д	КТ	ПЛАКК
	КЧ	0
	КЧ	—0
	НОП	
	КНВУ	МЛ, 1
	КЧ	000011000264В
	НОП	
	НОП	
А	КА	РАБ1; РАБ1 + 266В
	КА	КОН; 0
	КА	РАБ4; РАБ4 + 13В
	КА	0; 0
Д1	КТ	ЗНАРК
	КЧ	0
	КЧ	—0
	НОП	
	КНВУ	МЛ, 2
	КЧ	000010000240В
	НОП	
	НОП	

Этикетка	КОП	Адреса и замечания
А1	КА	РАБ2; РАБ2 + 242В
	КА	КОН1; 0
	КА	РАБ5; РАБ5 + 12В
	КА	0; 0
Д2	КТ	ПЛАПК
	КЧ	0
	КЧ	— 0
	НОП	
	КНВУ	МЛ, 3
	КЧ	000007000214В
	НОП	
	НОП	
А2	КА	РАБ3; РАБ3 + 216В
	КА	КОН2; 0
	КА	РАБ6; РАБ6 + 9
	КА	0; 0
ОТЫМЛ	ОПРЗ	ОТЫМЛ
	НОП	
	НОП	
Д3	КТ	СОПЗ
	КЧ	0
	КЧ	—0
	НОП	
	КНВУ	МЛ, 4
	КЧ	000013000334В
	НОП	
	НОП	
А3	КА	РАБ7; РАБ7 + 336В
	КА	0; 0
	КА	РАБ8; РАБ8 + 15В
	КА	0; 0
ВМЛ	ОПРЗ	ВМЛ
	НОП	
	НОП	
ЫМЛ	ОПРЗ	ЫМЛ

Этикетка	КОП	Адреса и замечания
	НОП	
	НОП	
ЧТЗ	ОПРЗ	ЧТЗ
	НОП	
	НОП	
ЗПЗ	ОПРЗ	ЗПЗ
	НОП	
	НОП	
ЗАЫ	ОПРЗ	ЗАЫ
	НОП	
	НОП	
СЧ	РЗВ	1
И1	ЗНАЧ	1
М	КИ	0; 1
М1	КИ	1; 0
И2	ЗНАЧ	2
ДЛ	КИ	13В; 0
СЧ1	РЗВ	1
КОНСТ	КЧ	+ 999999999Д
	КЧ	+ 16
	КЧ	+ 0
	КЧ	+ 7
	КЧ	+ 6
	КЧ	+ 3
	БАЗ	1; РАБ
РАБ1	РЗВ	300В
РАБП	РЗВ	10
ПР1	РЗВ	1
ПР2	РЗВ	1
ПР3	РЗВ	1
РАБ2	РЗВ	250В
РАБ4	РЗВ	20В
РАБ5	РЗВ	20В
РАБ3	РЗВ	230В
РАБ6	РЗВ	15В
РАБ7	РЗВ	350В
РАБ8	РЗВ	20В

ЗАКЛЮЧЕНИЕ

Бурное развитие вычислительной техники и расширение сфер ее применения предопределили повышенный интерес к вопросам программирования, так как без соответствующих программ вычислительные средства остаются бесполезными. Можно констатировать, что программирование становится широко распространенной профессией. Однако на данном этапе развития программирование еще только претендует на роль самостоятельной области науки. Существенное влияние на методы программирования оказывают особенности областей применения вычислительной техники. Так, можно говорить об инженерно-технических, научных, экономических, управленческих задачах, и каждая из названных областей имеет свои закономерности разработки и функционирования программ.

В этой книге рассмотрены только некоторые вопросы математического обеспечения АСУП. Ряд вопросов, имеющих существенное значение при разработке АСУП, не нашел достаточного отражения. Так, вопросы построения информационно-поисковых систем [6] имеют прямое отношение к АСУП, например, для организации поиска конструкторской и технологической документации. Кроме того, методы организации информационных массивов в информационно-поисковых системах можно использовать в системе обработки данных.

Общие вопросы программирования для цифровых вычислительных машин, некоторые подходы к построению систем программирования, в том числе и проблемно-ориентированных, изложены в [26, 35, 37, 38].

Некоторые алгоритмы обработки данных и вопросы организации информационных массивов приведены в [3, 18, 24, 42].

Для обеспечения информационной увязки задач и подсистем может плодотворно применяться теория графов [4].

Основные разделы математического обеспечения АСУП рассмотрены на примере ЭВМ «Минск-32». Кроме технической документации, в которой содержится полное описание этой машины

и внутреннего математического обеспечения, можно назвать пособие [23], в котором достаточно подробно изложены основы программирования на ЭВМ «Минск-32» на уровне системы символического кодирования. Весьма полезны в практической деятельности программистов сборники «Система математического обеспечения ЭВМ «Минск-32», выпускаемые институтом математики АН БССР.

Необходимо отметить, что в настоящее время начинают играть все большее значение машины третьего поколения, такие, как единая серия электронных вычислительных машин (ЕС ЭВМ). Принципы построения этих машин, архитектура, концепции операционных систем и основы программирования для них изложены в [8, 9, 31, 39] на примере классического представителя машин третьего поколения IBM/360.

Переход на массовую разработку АСУ предопределил актуальность вопросов типового проектирования. Можно сформулировать два принципиально разных подхода к реализации таких методов. При первом подходе необходимо построить модель системы управления, пригодную для некоторого класса объектов управления. В соответствии с выбранной целевой функцией управления для этой модели нужно разработать систему программирования, обеспечивающую создание математического обеспечения для конкретных объектов. Второй подход заключается в том, что для некоторого класса объектов управления определяется гипотетическая АСУ. Эта АСУ разделяется на подсистемы и задачи, для которых создается автономное математическое обеспечение, реализующее соответствующую обработку информации. При создании АСУ для конкретного объекта информационную увязку отдельных модулей, их согласование можно реализовать с помощью специального генератора программ. Работы по реализации обоих направлений проводятся как в нашей стране, так и за рубежом.

ЛИТЕРАТУРА

1. Алгоритмический язык АЛГОЛ-60. Пер. с англ. М., 1965.
2. Алгоритмический язык АЛГОЛ-68. Пер. с англ. Под общ. ред. А. П. Ершова. «Кибернетика», 1969, № 6, 1970, № 1.
3. *Алферова З. В., Волович М. А.* Сортировка информации с помощью электронных вычислительных машин. М., 1965.
4. *Алферова З. В., Ежева В. П.* Применение теории графов в экономических расчетах. М., 1971.
5. *Беллман Р., Дрейфус С.* Прикладные задачи динамического программирования. М., 1965.
6. *Белоногов Г. Г., Котов Р. Г.* Автоматизированные информационно-поисковые системы. М., 1968.
7. *Брандон Д. Х.* Организация работы на вычислительном центре. М., 1970.
8. Вычислительная система *IBM*/360. Принципы работы. Под ред. В. С. Штаркмана. Пер. с англ. М., 1969.
9. *Джермейн К.* Программирование на *IBM*/360. Пер. с англ. Под ред. В. С. Штаркмана. М., 1971.
10. *Дубовицкая Р. К. и др.* Система автоматической обработки данных на базе языка КОБОЛ. М., 1971.
11. *Дудкин Г. Е., Хотяшов Э. Н.* Система организации постоянной информации предприятия при использовании ЭВМ. М., 1971.
12. *Емеличев В. А., Комлик В. И.* Решение задач дискретного программирования методом построения последовательности планов. ДАН СССР, т. 188, 1969, № 2.
13. *Емеличев В. А., Комлик В. И.* Применение динамического программирования к решению задачи размещения. Докл. АН БССР, 1966, № 10.
14. *Ершов А. П.* Программирование за рубежом. Труды 2-й Всесоюзной конференции по программированию. Новосибирск, 1970.
15. *Зуховицкий С. И., Авдеева Л. И.* Линейное и выпуклое программирование. М., 1964.
16. *Иньков Ю. И.* Электронная вычислительная техника и капиталистическая экономика. М., 1968.
17. *Камынин С. С., Любимский Э. З.* Алгоритмический машинно-ориентированный язык АЛМО.— В сб.: «Алгоритмы и алгоритмические языки». М., 1968.
18. *Китов А. И.* Программирование информационно-логических задач. М., 1967.
19. *Королев М. А.* Обработка экономической информации на ЭВМ. М., 1964.
20. *Королев М. А. и др.* Сообщение об алгоритмическом языке АЛГЭК. «Кибернетика», 1966, № 2.
21. *Комлик В. И., Емеличев В. А.* Решение задачи целочисленного программирования с переменными 0—1. Докл. АН БССР, 11, 1967, № 12.
22. *Корбут А. А., Финкельштейн Ю. Ю.* Дискретное программирование. М., 1969.
23. *Кушнерев Н. Т., Неменман М. Е., Цагельский В. И.* Программирование для ЭВМ «Минск-32». М., 1972.
24. *Лавров С. С., Гончарова Л. И.* Лекции по автоматической обработке данных. Ч. 1. Организация информационных массивов. М., 1969.

25. *Лавров С. С.* Универсальный язык програмирования (АЛГОЛ-60). М., 1964.
26. *Ледли Р.* Программирование и использование цифровых вычислительных машин. Пер. с англ. М., 1966.
27. Общеотраслевые руководящие методические материалы по созданию АСУП. Минск, 1972.
28. Описание языка АЛГАМС.— В сб.: «Алгоритмы и алгоритмические языки». Вып. 3. М., 1968.
29. Отраслевой РТМ. Автоматизированные системы управления (АСУ). Обработка данных и программирование. Обозначения условные графические. РТМ 25—8—70. М., 1970.
30. *Пржиялковский В. В., Смирнов Г. Д., Пыхтин В. Я.* Электронная вычислительная машина «Машина-32». М., 1972.
31. Принципы построения современных операционных систем (обзор). М., 1969.
32. Расчет экономической эффективности внедрения ЭВМ. М., 1968.
33. *Рафаэль В. и др.* Краткий обзор по алгоритмическим языкам для символьных и алгебраических преобразований. Пер. с англ. М., 1969.
34. Система программ обработки экономической информации на ЭВМ «Минск-32». Под ред. В. С. Крисевича и Э. Н. Хотяшова. Минск, 1972.
35. *Тамм Б. Г.* Некоторые концепции системного программирования. Изв. АН Эст. ССР, № 1. Таллин, 1970.
36. Универсальный язык программирования *PL*/1. Пер. с англ. Под ред. В. М. Курочкина. М., 1968.
37. *Фишер Ф. П., Суиндл Д. Ф.* Системы программирования. Пер. с англ. М., 1971.
38. *Флорес А.* Программное обеспечение. М., 1971.
39. Функциональная структура *OS*/360. Пер. с англ. Под ред. В. С. Штаркмана. М., 1971.
40. *Хотяшов Э. Н. и др.* Система обработки экономической информации на ЭВМ «Минск-22». М., 1969.
41. *Шкурба В. В. и др.* Задачи календарного планирования и методы их решения. Киев, 1966.
42. *Шураков В. В.* Математическое обеспечение ЭВМ. Киев, 1971.
43. *Land A. H., Doig A. G.* An automatic method of solving discrete programming problems. «Econometrica», 1960, 28, № 3.
44. *Samuel W. Reynolds. A.* Generalized Polyphase Merge Algoritm. «Communications of ACM», 1963, № 5.

ОГЛАВЛЕНИЕ

Хотяшов Эрнст Никифорович

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ АСУ

Редактор А. Белянкина
Худож. редактор В. Валентович
Техн. редактор П. Фрайман
Корректор Л. Синегрибова

АТ 11572. Сдано в набор 10/IX 1973 г. Подписано в печать 26/VI 1974 г. Бумага 60×90¹/₁₆ типогр. № 3. Печ. л. 22. Уч.-изд. л. 23,46. Изд. № 72-94. Зак. 440. Тираж 10 000 экз. Цена 1 руб. 30 коп.

Издательство «Вышэйшая школа» Государственного комитета Совета Министров БССР по делам издательств, полиграфии и книжной торговли. Редакция литературы по математике, физике и энергетике. 220600. Минск, ул. Кирова, 24.

Полиграфический комбинат им. Я. Коласа Государственного комитета Совета Министров БССР по делам издательств, полиграфии и книжной торговли. Минск, ул. Красная, 23.

1 р. 30 к.

38459

ИЗДАТЕЛЬСТВО „ВЫШЭЙШАЯ ШКОЛА“