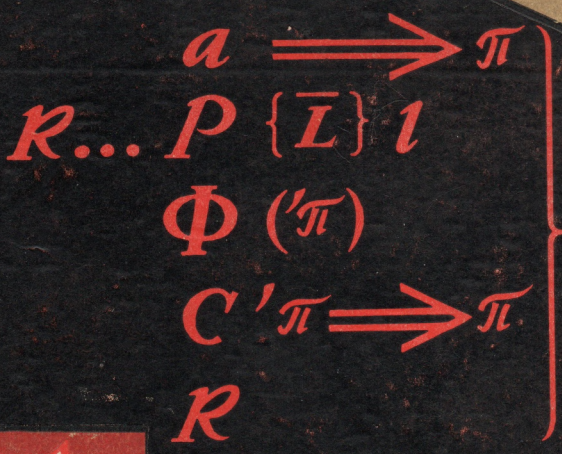
Е. Л. Ющенко



АДРЕСНОЕ ПРОГРАММИРОВАНИЕ

Е. Л. Ющенко

Адресное программирование

Государственное издательство
ТЕХНИЧЕСКОЙ ЛИТЕРАТУРЫ УССР
Киев—1963

6П2.15
Ю98

681.142.2

В книге изложен универсальный алгоритмический адресный язык и основанный на нем метод программирования на ЭВЦМ.

Адресный язык пригоден для описания арифметических и сложных информационно-логических задач (программирующие программы, задачи экономической кибернетики или распознавания образов), для описания электронных вычислительных цифровых машин как автоматов с программным управлением и может быть положен в основу построения системы автопрограммирования. Метод адресного программирования не связан с конкретными особенностями машин и может служить общим методом составления программ для ЭВЦМ.

Книга рассчитана на инженеров и научных сотрудников, работающих в области вычислительной математики и техники, а также специалистов других областей и может быть полезна студентам, изучающим курс вычислительных машин и программирования.

Рецензент *В. С. Чернявский*, канд. физ.-мат. наук

Редакция литературы по вопросам энергетики, радио и телевидения.
Заведующий редакцией инж. *М. Г. Писаренко*

ПРЕДИСЛОВИЕ

С появлением электронных вычислительных цифровых машин (ЭВЦМ) с программным управлением возник новый прикладной раздел современной теории алгоритмов — разработка способов точного описания алгоритмов, реализуемых этими машинами,— называемый программированием для ЭВЦМ. Сфера использования ЭВЦМ с каждым днем расширяется и охватывает все новые и новые области деятельности человека, среди которых прежде всего должна быть названа кибернетика.

Для успешного внедрения ЭВЦМ необходима упорная и систематическая работа по созданию стройной теории программирования и по изучению и составлению математических описаний (моделей) различных процессов, подлежащих автоматизации. Для этих исследований требуется специальная подготовка как от математиков, занимающихся вопросами теории и практики программирования, так и от физиков, биологов, техников, экономистов, психологов, лингвистов, желающих использовать в своей работе ЭВЦМ, не прибегая к помощи посредника-программиста.

Однако в существующих книгах по программированию, как правило, на основе языка конкретной или некоторой условной машины с фиксированными адресностью и набором операций излагаются основы ручного программирования. В данной книге сделана попытка изложить общие правила программирования безотносительно к конкретным особенностям тех или иных машин — общий метод составления программ для ЭВЦМ с упором на автоматическое программирование.

В основу книги положен разработанный автором язык, названный адресным и используемый в качестве входного

языка на некоторых ЭВЦМ. Идея создания нового алгоритмического языка возникла в результате работы семинара по теории алгоритмов (1957—1958 гг.), руководимого проф. В. М. Глушковым, проф. Л. А. Калужниным, канд. физ.-мат. наук В. С. Королюком и автором. Первый вариант языка был предложен автором совместно с В. С. Королюком в 1958 г.

Адресный язык может быть использован для описания не только арифметических задач, но и для сложных информационно-логических алгоритмов таких, как программирующие программы, задачи экономической кибернетики, задачи, связанные с проблемой распознавания образов, и др. Примеры, иллюстрирующие эти возможности, читатель найдет в гл. IV, V, VII. В гл. VI на примере машины «Урал» показана возможность применения адресного языка для описания ЭВЦМ как автоматов с программным управлением.

Для первого ознакомления можно рекомендовать гл. I, II, III, IV, VI. При этом параграфы, отмеченные звездочкой, могут быть опущены. Читатель, имеющий некоторую подготовку по программированию, может на основе гл. I, III, V, VII обобщить и расширить свои знания, ознакомиться с принципами автоматического программирования. Лица, интересующиеся вопросами теории программирования, найдут в книге постановку задач, для решения которых требуется дальнейшее развитие предлагаемого метода программирования (гл. III, V, VII)

Изложение иллюстрируется многими примерами и упражнениями. Отдельная глава посвящена разбору более сложных задач и алгоритмов их решения. Это дает возможность читателю приобрести твердые практические знания и навыки по программированию.

Все замечания читателей, которые будут приняты автором с благодарностью, просим направлять по адресу: *Киев, 4, Пушкинская, 28, Гостехиздат УССР.*

ГЛАВА I

АДРЕСНЫЙ ЯЗЫК

1. ПОНЯТИЕ АЛГОРИТМА

Понятие алгоритма — одно из основных в математике.

Под алгоритмом понимают точное предписание о выполнении в строго установленном порядке определенной системы операций, дающее решения всех задач некоторого класса.

Обычно алгоритмы задаются в виде словесных предписаний, различных формул и схем. Например, корни квадратного уравнения через его коэффициенты выражаются известной формулой. Алгоритм Евклида для определения наибольшего общего делителя двух многочленов f и g может быть записан в следующей словесной форме. Делим f на g; остаток и частное, получаемые при делении, обозначаем через g_1 и a_1. Затем делим g на остаток g_1 (когда $g_1 \neq 0$); в результате получаем второй остаток g_2 и частное a_2 и т. д. Наконец, получим остаток g_n, на который полностью разделится предыдущий остаток g_{n-1}. Этот последний остаток и является наибольшим общим делителем исходных многочленов.

В приведенном примере, как и в любом алгоритме, предписываемые операции осуществляются не над числами или многочленами, а над их записью: в каждом отдельном случае рассматриваются определенные знаки, к которым применяются правила их преобразования.

Запись условий задач, промежуточных и окончательных результатов, а также самих алгоритмов представляет задание определенного рода информации. Простейшие знаки, с помощью которых осуществляется запись какой-либо информации, называются буквами. Буквы должны удовлетворять единственному условию — различимости,

т. е. для любых двух букв можно установить, различны они или одинаковы. Для записи информации употребляется тот или иной набор букв; фиксированные (конечные) наборы букв называют алфавитами, а конечные упорядоченные последовательности букв — словами в данном алфавите. Удобно считать, что в любой алфавит входит «пустая» буква, которую обычно используют для разделения слов.

Для записи целых положительных чисел в десятичной системе счисления используют десять букв: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Присоединение к указанным десяти буквам еще двух — запятой и знака минус позволяет записывать десятичные числа (дробные, целые, положительные и отрицательные). Алфавит из одной буквы — вертикальной черты — дает возможность записывать числа натурального ряда. Число единица в этом алфавите запишется как |, а число пять как ||||| и т. д. Для записи рациональных чисел может быть использован алфавит, состоящий из трех букв-черточек — вертикальной, горизонтальной и наклонной: | , —, /. Тогда, например, дробь 1/3 запишется как |/|.|.

Каковы бы ни были объекты, к которым требуется применить алгоритм, всегда можно выбрать подходящий алфавит и сформулировать определенные правила, позволяющие записывать в нем объект определенным словом. Другими словами, всякая информация представляет собой слово в определенном алфавите.

Для многих рассуждений удобно словам в заданном фиксированном алфавите ставить в соответствие определенные слова в базисном алфавите из двух букв. Это может быть сделано, например, следующим способом, предложенным А. А. Марковым.

Обозначим буквы исходного n-буквенного алфавита через $a_1, a_2, \ldots, a_n$ и базисного двубуквенного алфавита через α, β и введем следующее соответствие:

$$a_1 = \beta\alpha\beta$$
$$a_2 = \beta\alpha\alpha\beta$$
$$\ldots\ldots$$
$$a_n = \beta\alpha\alpha\ldots\alpha\beta.$$

Тогда, например, слову $a_2a_3a_1$ однозначно соответствует слово двубуквенного алфавита $\beta\alpha\alpha\beta\beta\alpha\alpha\alpha\beta\beta\alpha\beta$, и наоборот.

Возможность кодирования любой информации в двубуквенном алфавите положена в основу выполнения операций в машинах, построенных на элементах с двумя устойчивыми состояниями, каждому из которых соответствует одна из букв двубуквенного алфавита. (Обычно в качестве этих букв берут знаки 0 и 1).

При формальном описании алгоритмов существенную роль играет, с одной стороны, выбор способа записи (кодирования) информации и, с другой,— способов описания самих алгоритмических действий. Естественно ставить вопрос о записи информации в виде, позволяющем в известном смысле наиболее просто формулировать алгоритм ее переработки. Однако в связи с необходимостью хранения информации в запоминающих устройствах машин интересны также наиболее экономные способы кодирования, с помощью которых необходимая информация представляется в наиболее сжатой форме. С теоретической точки зрения важно отметить, что уже двубуквенный алфавит достаточен для кодирования произвольной информации.

Вопросы о способах описания алгоритмов для решения тех или иных классов задач, что такое, вообще, задача и ее решение, как следует осуществлять решение задач, каковы общие встречающиеся при этом закономерности, как сравнивать различные вычисления, когда одно вычисление следует считать более удобным, чем другое, и т. д., составляют предмет рассмотрения теории алгоритмов. Точнее говоря, теория алгоритмов, рассматривая класс объектов, удовлетворяющих некоторому точному определению понятия алгоритма, занимается задачами двоякого рода: во-первых, описывает индивидуальные элементы указанного класса, т. е. создает новые алгоритмы; во-вторых, доказывает положения, относящиеся к классу в целом. Возникновение этой науки наряду с большой практической значимостью играет важную познавательную роль.

В теории алгоритмов считается, что понятие алгоритма не должно исчерпываться алгоритмами, содержание которых сводится к вычислениям над числами. Это понятие должно быть обобщено, поскольку с теоретической точки зрения нет никакой принципиальной разницы, например, между вычислениями арифметическими или алгебраическими и составлением графика движения поездов, если эта работа полностью формализована. Таким образом,

понятие вычислений может быть расширено: как вычисления можно рассматривать не только действия с числами, многочленами, неопределенными интегралами и т. д., но и перевод с одного языка на другой, составление расписаний и другие задачи. Принципиального различия в осуществлении таких алгоритмов не существует. Правда, в одних случаях алгоритмы сформулированы точно, а в других — либо в общих чертах и полностью не формализованы и поэтому не подготовлены для постановки на машинах (такова, например, выигрышная стратегия игры в шахматы), либо вовсе еще не сформулированы (например, алгоритм управления нефтеперерабатывающим заводом).

Поэтому понятие алгоритма может быть определено несколько шире: *алгоритм есть система правил, по которой совершается определенное преобразование некоторой информации.*

Правила преобразования информации в различных алгоритмах весьма разнообразны и качественно различны, в связи с чем попытка построения машины, реализующей произвольные алгоритмы, на первый взгляд может показаться безнадежной. Однако в теории алгоритмов выясняется, что, несмотря на все разнообразие правил преобразования информации, все конкретные алгоритмы могут быть составлены из весьма небольшого числа элементарных правил.

Наборы правил, из которых могут быть построены произвольные алгоритмы, называют алгоритмически полными; выбор таких наборов можно осуществить различными способами. В случае кодирования исходной информации в цифровом коде алгоритмически полную систему образуют, например, три операции ЭВЦМ (электронных вычислительных цифровых машин): переадресация команд (изменение адресов команд), условный переход (передача управления к одной или другой команде в зависимости от реализации некоторого условия) и сдвиг (сдвиг кода на заданное число разрядов вправо или влево).

В действительности, универсальные цифровые машины имеют наборы операций, значительно превышающие необходимый минимум, что обеспечивает, наряду с принципиальной возможностью реализации на них произвольных алгоритмов, их практическую приемлемость для этих целей. Этим и объясняется тот факт, что современные

вычислительные цифровые машины широко применяются для преобразований информации, весьма далеких от вычислений в собственном смысле.

В настоящее время можно считать установившимся следующее представление о построении алгоритмической теории, предложенное А. Н. Колмогоровым [8]:

1. Рассматривается система объектов, называемых условиями, или состояниями.

2. Определяется совокупность алгоритмических (допустимых) действий — преобразований состояний. Результат отдельного действия составляет переход от одного состояния системы к другому.

3. Алгоритмический процесс составляет последовательность допустимых преобразований. На каждом этапе происходит преобразование состояния, полученного в результате применения предыдущего алгоритмического этапа.

4. Алгоритм определяется точным указанием порядка применения допустимых преобразований: это означает, что алгоритм может быть сообщен любому другому лицу в виде конечного перечня указаний и повторен им как в любое время, так и произвольное число раз, независимо от того, знакомо ли это лицо со смыслом решаемой задачи; результат применения алгоритма будет определяться лишь условием задачи.

5. Решением задачи является заключительное состояние, которое достигается в результате применения алгоритма к исходному состоянию.

В дальнейшем будем рассматривать алгоритмы, удовлетворяющие свойству конечности. Это значит, что после конечного (возможно, очень большого) числа шагов применения алгоритма следует вполне определенный результат.

Способы записи рассматриваемых объектов и преобразования этих записей могут быть различными. Разработанные в теории алгоритмов системы записей и их преобразований называются определенными системами алгоритмов. Такую систему алгоритмов образуют так называемые алгоритмы Маркова, машины Тьюринга, системы программирования для универсальных автоматических машин с программным управлением. Предлагаемый автором метод программирования, единый для широкого класса универсальных математических машин, также представляет собой некоторую алгоритмическую систему.

2. ОСНОВНЫЕ ПОНЯТИЯ И СРЕДСТВА АДРЕСНОГО ЯЗЫКА

Адресный язык весьма близок к обычному языку математических формул. В нем приняты все символы, которыми в математике обозначают величины, векторы, функции, множества, т. е. скобки, цифры, буквы с индексами и без них и т. д., а также все знаки математических операций ($+$, $-$, $\times$, $:$, $\sqrt{}$, ln, sin и т. д.). Значение этих символов обычное, но иногда на них накладываются определенные ограничения, определяемые рассматриваемой задачей и не связанные с адресным языком; тем самым их значение может быть изменено. Однако в адресный язык вводятся специальные понятия и соответствующие символы, благодаря которым он становится более приспособленным для описания алгоритмов.

Перейдем к определению основных понятий и описанию основных выразительных средств адресного языка.

Строки. Адресная запись алгоритма состоит из строк. В каждой из них записывается одно или несколько алгоритмических действий. Запись каждого действия называется формулой. Если в строке имеется несколько формул, то между ними ставится точка с запятой или запятая. При записи строк на бумаге их пишут одну под другой; тем самым определяются границы строки. При линейной записи строк между ними ставится точка. Обычно действия адресного алгоритма выполняются одно за другим в порядке записи строк, но этот порядок может быть изменен с помощью некоторых из допустимых в языке формул.

Штрих-операция и адресное отображение. Одним из основных понятий адресного языка является штрих-операция, задающая отображение некоторого множества A, называемого множеством адресов, на множество B, называемое множеством содержимых этих адресов. Указанное отображение называют адресным. Символ операции $'$ (штрих) ставится вверху слева от аргумента

$$'a = b,$$

где a — аргумент ($a \in A$);

b — результат операции ($b \in B$).

Читаем: «штрих a равно b».

Таким образом, штрих-операция определяет некоторую функцию одного аргумента — штрих-функцию. Аргумент

штрих-функции называется адресом, ее значение — содержимым адреса. Приведенное выражение можно читать и так: «a — адрес b», или «a содержит b», или «b содержимое (или содержится по) a». Все эти выражения равнозначны.

Пусть даны множество A адресов a_i и множество B содержимых b_i. Штрих-функция будет задана, если каждому адресу a из множества A поставлено в соответствие некоторое содержимое b из множества B. Понятия «адрес» и «содержимое адреса» нужно рассматривать как неопределяемые термины.

Штрих-функция однозначна, т. е. одному адресу соответствует только одно содержимое. Обратное отображение не обязательно однозначно, — одно и то же содержимое может отвечать нескольким различным адресам.

Штрих-операция считается алгоритмически выполнимой, т. е. по адресу всегда можно узнать его содержимое (извлечь содержимое по адресу).

В абстрактных рассмотрениях множества A и B могут иметь любую природу и не обязаны быть непременно числовыми множествами, как в естественных машинных языках, т. е. штрих-функция $'$ может быть определена на произвольном множестве аргументов A и иметь значения в произвольном конечном множестве B.

Для обозначения элементов множества адресов используются буквы некоторого исходного алфавита с индексами или без них, натуральные числа и слова из букв исходного алфавита и цифр. Примем в качестве исходного алфавита объединение латинского, русского и греческого алфавитов. Применение последних в языке будет объясняться по мере надобности. Таким образом, для обозначения адресов могут быть использованы, например, записи: α_2, 1012, $\alpha 3$, сумма 1, ω.

Если a не принадлежит множеству адресов A ($a \notin A$), то выражение $'a$ не имеет смысла. Будем предполагать, что операция $'$ (штрих) применима к тем элементам, которые используются в качестве ее аргументов.

Повторное применение операции $'$ (штрих) приводит к понятию адреса второго ранга. Так, если $'a_1 = a_2$ и $a_2 \in A$, т. е. оно, в свою очередь, является адресом, то имеется такое b, что $'a_2 = b$. Это символически записывается так:

$$^2a_1 = '('a_1) = ''a_1 = 'a_2 = b.$$

В этом случае говорят: «a_1 — адрес второго ранга для b», или «a_1 — адрес адреса b», или «a_1 — фиксатор b».

Аналогично вводится понятие адресов более высокого ранга. Так, запись ${}^4a = b$ означает, что имеются элементы x_1, x_2, $x_3 \in A$ такие, что ${}'a = x_1$; ${}'x_1 = x_2$; ${}'x_2 = x_3$; ${}'x_3 = b$, т. е.

$$ {}^4a = {}^3x_1 = {}^2x_2 = {}'x_3 = b. $$

Условимся называть b адресом нулевого ранга элемента b. Существование адресов высшего ранга зависит от заданного отображения множества B на множество A. Если в множестве B есть подмножество B_1, принадлежащее A, и A_1 — множество адресов элементов из B_1, (если ${}'a_1 = b_1$, где $b_1 \in B_1$, то $a_1 \in A_1$), то A_1 будет множеством адресов второго ранга. Пусть B_0 то подмножество множества B, элементы которого являются содержимым элементов из B_1. Тогда устанавливается соответствие между множествами A_1 и B_0, определяющее операцию «двойная штриховка». Символически ${}^2a_1 = b_0$, где $a_1 \in A_1$, $b_0 \in B_0$.

Подобным же образом можно выделить и соответствия, определяющие адреса высших рангов. Все они однозначно определяются исходным отображением B на A.

Операция следования. Для упорядочения элементов в множествах вводится операция следования, обозначаемая символом C. Пусть α — элемент множества, упорядоченного операцией следования C, тогда в общем случае $C\alpha$ означает элемент, следующий за α, а $CC\alpha$ — элемент, следующий за элементом $C\alpha$, и т. д. $C^{-1}\alpha$ означает элемент, предшествующий элементу α.

Вводится сокращение

$$ C^0\alpha = \alpha;\ C^2\alpha = CC\alpha;\ \ldots\ ;\ C^{-1}C^{-1}\alpha = C^{-2}\alpha \text{ и т. д.} $$

В каждом конкретном случае операция следования должна описываться алгоритмически. Например, в множестве натуральных чисел операция следования может быть определена как операция прибавления единицы: $Cn = n + 1$.

Элемент α, для которого $C^{-1}\alpha$ не определен, называется первым, соответственно, элемент β, для которого $C\beta$ не определен — последним элементом множества.

Можно рассматривать множества с несколькими операциями следования. Например, для элементов квадратной матрицы $A\{a_{ij}\}$ ($i, j = 1, 2, \ldots, n$) естественно рас-

сматривать две операции следования, определяющие по элементу a_{ij} элемент a_{ij+1} (следование по строкам) и элемент $a_{i+1,j}$ (следование по столбцам).

Пусть задано адресное отображение элементов матрицы A на множество адресов, упорядоченное операцией следования C так, что

$$
\begin{aligned}
a_{11} &= {}'\alpha_0;\\
a_{12} &= {}'C\alpha_0;\\
&\vdots\\
a_{1n} &= {}'C^{n-1}\alpha_0;\\
&\vdots\\
a^{ij} &= {}'C^{n(i-1)+j-1}\alpha_0;\\
&\vdots\\
a_{nn} &= {}'C^{n^2-1}\alpha_0.
\end{aligned}
$$

Тогда операции следования C_1 и C_2 в множестве элементов матрицы A могут быть описаны с помощью операций следования, установленных в множестве их адресов C и C^n, т. е.

$$
\begin{gathered}
a_{ij} = {}'\alpha_{ij}; \quad \alpha_{ij} = C^{n(i-1)+j-1}\alpha_0;\\
C_1 a_{ij} = a_{i,\,j+1} = {}'C\alpha_{ij};\\
C_2 a_{ij} = a_{i+1,\,j} = {}'C^n\alpha_{ij}.
\end{gathered}
$$

Операция засылки. Для определения и изменения адресного отображения применяется операция засылки, символ которой $\Rightarrow$. Запись операции

$$b \Rightarrow a \tag{1.1}$$

(читается так: «b заслать по адресу a») означает, что:

1) элемент a включается в множество адресов A;

2) элемент b включается в множество содержимых B;

3) устанавливается соответствие ${}'a = b$;

4) все ранее установленные соответствия вида ${}'x = y$, где $x \neq a$, остаются неизменными.

Таким образом, после засылки $b \Rightarrow a$ изменяется значение всех выражений ${}'a$, ${}^2a$, ${}^3a$ и т. д., при этом ${}'a$ после засылки равно значению b до засылки (это необходимо учитывать, если в выражение b входит ${}'a$, ${}^2a$, ... и т. д.).

Адресное отображение можно установить, сделав множество засылок вида

$$b_{i_j} \Rightarrow a_j,$$

где a_j пробегает все значения из A по одному разу; b_{i_j} принимает такие значения из B, что $'a_j = b_{i_j}$, причем одно значение может повторяться несколько раз.

Справа и слева от знака $\Rightarrow$ в формуле (1.1) может стоять адрес любого ранга $^n\alpha$ или выражение $C^n\alpha$. В этом случае выполнение операции засылки состоит в том, что извлекаются содержимые (слева и справа от знака $\Rightarrow$) до тех пор, пока над элементами уже не будет стоять знака штрих, затем при наличии символа C определяется следующий элемент и, наконец, производится соответствующая засылка.

Все описанные операции считаются алгоритмически выполнимыми. Включение в набор алгоритмических операций штрих-операции и засылки и, вообще, введение понятия адресного отображения отражает специфику решения задач на ЭВЦМ с программным управлением.

Для обеспечения массовости алгоритма необходимо в нем указывать не условия конкретной задачи, а лишь их символическое обозначение. При описании алгоритмов в машинных кодах в качестве таких символических обозначений используются адреса, по которым в памяти машины размещаются соответствующие элементы информации.

Элементы исходной информации и адреса, используемые для записей алгоритмов, кодируются в числовых (обычно двоичных) кодах. Поэтому естественно рассматривать элементы информации и их символические адреса как элементы одного и того же конструктивного множества слов, используемых для кодирования информации, и производить действия над этими элементами, независимо от того, являются ли они адресами или элементами информации. Размещение этих символов по адресам запоминающих устройств машины определяет конкретное адресное отображение.

Метки и меченые строки. Те или иные строки адресного алгоритма для указания порядка их следования могут отмечаться метками. Такие строки называются ме-

ными. Меткой может быть цифра, буква, слово из цифр (целое число), слово или несколько слов из цифр и букв, с индексами или без них.

Например:

214, A31, K_1, *Соколов* 1.

Метки могут, вообще говоря, выбираться почти произвольно. Допускаются составные метки вида $M \circ N$ (где $\circ$ — специальный знак, M, N — метки). Метка с последующим троеточием ставится слева от отмечаемой строки. Троеточие в записи алгоритма всегда означает, что слева от него стоит метка, отмечающая данную строку. Строки могут отмечаться несколькими метками, после каждой из которых ставится троеточие.

Метки безусловного перехода. Отдельную строку может составлять метка без последующего троеточия. В этом случае она называется меткой безусловного перехода. При реализации алгоритма выполнение такой строки состоит в переходе к строке, помеченной той же меткой (или имеющей в качестве одной из своих меток данную). Таким образом, составляющая строку метка имеет смысл только тогда, когда в алгоритме имеется другая строка, помеченная той же меткой.

Отдельную строку алгоритма (без последующего троеточия) может составлять и адрес любого ранга. В этом случае содержимое его по данному рангу является меткой. Выполнение такой строки состоит в извлечении содержимого по данному рангу и в переходе к строке алгоритма, помеченной меткой, определяющейся этим содержимым.

Формула останова. Формула останова обозначается символом ! и может составлять отдельную строку алгоритма. Выполнение формулы останова означает окончание алгоритмического процесса.

Основные средства адресного языка. Все перечисленные в этом параграфе средства адресного языка, как будет показано далее, при наличии бесконечной линейной упорядоченной памяти, достаточны для записи любого алгоритма, который может быть описан какими-либо другими средствами. Оказывается даже, что достаточно употреблять операцию извлечения содержимого из адреса не выше второго ранга.

3. АДРЕСНАЯ ФУНКЦИЯ

Расширим общепринятое понятие функции и соответствующего ей выражения, построенного из символов одноместных и двухместных операций[1], скобок и величин, включив в качестве одной из допустимых элементарных операций штрих-функцию. Такие функции называют адресными. Частным случаем адресной функции является любая функция, выражение которой не содержит знака штрих-функции.

Выражения адресных функций в языке используются для указания самостоятельных алгоритмических действий — формул или меток безусловного перехода, а также для составления записей других алгоритмических действий — формул засылки, обмена, предикатных, вхождения, относительного перехода, циклирования.

Введение понятия адресной функции накладывает ограничения на множество адресов и множество содержимых. Адресная функция может иметь смысл только тогда, когда к адресам и содержимым применимы операции, употребленные в записи функции. Как правило, будем пользоваться знаками арифметических действий, но в некоторых случаях будут употребляться и другие символы операций как двухместных, так и одноместных (в этих случаях смысл их будет специально оговорен).

Например, следующие выражения будут записями адресных функций:

$$'a$$

$$^2b + 'c - '311$$

$$311 + '311$$

$$'('x + 'd) - \sqrt{'('a + 'b)} + \sin 'd,$$

причем требуется, чтобы a, b, $'b$, c, 311, $'x + 'd$, $'a + 'b$, d, x принадлежали множеству адресов A.

Если определяющее штрих-функцию отображение задано, то для любой адресной функции можно найти ее значение. Для этого нужно выполнить все действия, указанные в выражении функции (в том числе и извлечение содержимых по адресу), и получить выражение, не содер-

---

[1] k-местной операцией называется операция, определяющая функцию k аргументов.

жащее символов операций. Так, если $'b = d$, $'c = 14$, $'14 = 18$, $'d = 1$, то адресная функция

$$'d \times 'c + ''b + ''c$$

равна 33.

Однако адресная функция может быть написана формально, т. е. без предварительного задания отображения множества B на множество A, определяющего смысл штрих-операции. В этом случае адресная функция символизирует операции над пока еще неизвестным содержимым выражений, входящих в нее под символом $'$ (штрих). Если задать отображение, определяющее штрих-операцию, причем в множество A входят все символы, стоящие в функции под этим знаком, то функция приобретает некоторое вполне определенное значение. Таким образом, значение одной и той же адресной функции может быть различным в зависимости от отображения, определяющего штрих-операцию. Фиксированное адресное отображение однозначно определяет значение любой адресной функции.

Однако ограничения, накладываемые совместным применением штрих-операции и знаков действий, не следует преувеличивать. Так, вовсе не обязательно, чтобы над адресами можно было производить те или иные математические действия. Если в рассматриваемые выражения символы адресов входят всегда под знаком $'$, то достаточно, чтобы действия можно было производить над содержимыми адресов. Наоборот, если перед сложным выражением стоит знак $'$, то вовсе не обязательно, чтобы все входящие в выражение символы были адресами. Достаточно, чтобы значение выражения было адресом.

Адреса должны быть числами, если к ним применяются арифметические операции. Кроме того, их можно считать переменными, обозначаемыми буквами или другими символами и принимающими числовые значения. Таким образом, значение адресной функции зависит от значений входящих в нее переменных. Однако в любом алгоритме, при решении отдельной конкретной задачи, значения всех употребляемых переменных определяются ее условиями. Поэтому можно считать, что значение адресной функции в каждый данный момент зависит только от адресного отображения в этот момент.

Введем понятие ранга R адресной функции. Ранг адреса n-ранга равен n, т. е.

$$R({}^{n}a) = n.$$

Для адресных функций это понятие обобщается следующим образом [21].

1. Адресная функция a, не содержащая штрих-операции, имеет ранг R, равный нулю, т. е.

$$R(a) = 0.$$

2. Пусть ранг функции f $R(f) = s$, тогда ранг функции $'f$

$$R('f) = R(f) + 1 = s + 1.$$

3. Любая функция от одного переменного x, отличная от штрих-операции, имеет ранг, равный рангу ее аргумента, т. е.

$$R(O_1(x)) = R(x).$$

4. Пусть $R(a) = s_1$, $R(b) = s_2$. Ранг функции $a\theta b$ (где θ — любая арифметическая или логическая операция от двух аргументов) равен максимальному из рангов s_1 и s_2, т. е.

$$R = (a\theta b) = \max[R(a),\ R(b)].$$

Произвольный адресный алгоритм может быть представлен в эквивалентной ему (в содержательном смысле) форме, в которой все входящие в запись алгоритма адресные функции имеют ранг не выше второго.

Например, адресные формулы

$${}^{n}\alpha \Rightarrow \beta \quad (n \geqslant 3)$$

или

$$a \Rightarrow {}^{m}b \quad (m \geqslant 2)$$

могут быть записаны соответственно в виде

$$\left.\begin{array}{l} {}^{2}\alpha \Rightarrow \beta \\ {}^{2}\beta \Rightarrow \beta \\ \cdot \\ \cdot \\ \cdot \\ {}^{2}\beta \Rightarrow \beta \\ {}^{2}\beta \Rightarrow \beta \end{array}\right\} n-2 \text{ раза} \qquad \left.\begin{array}{l} {}^{2}b \Rightarrow b \\ {}^{2}b \Rightarrow b \\ \cdot \\ \cdot \\ \cdot \\ {}^{2}b \Rightarrow b \\ a \Rightarrow {}'b \end{array}\right\} m-1 \text{ раз.}$$

Кроме того, любая адресная формула

$$a \Rightarrow b$$

(где a и b — адресные функции соответственно ранга n и m) может быть заменена эквивалентной ей программой из двух строк

$$a \Rightarrow \omega$$
$$'\omega \Rightarrow b,$$

где ω — некоторый адрес, содержимое которого на выполнение алгоритма не влияет; такой адрес называют рабочим, или свободным относительно данного алгоритма.

По приведенному определению можно построить адресную функцию любого ранга, по записи заданной функции — найти ее ранг.

4. ДОПУСТИМЫЕ ФОРМУЛЫ

Адресный язык предназначен для алгоритмизации задач и программирования на вычислительных машинах. Поэтому, несмотря на полноту его основных средств, для удобства записи в нем алгоритмов и их обозримости в адресный язык введены понятие адресов любого ранга и перечисленные далее выразительные средства.

Применяя все средства адресного языка, можно получить алгоритмы настолько отличные от записанных с помощью его основных средств, что пути преобразования одних в другие обычно неясны. Однако термин «основные средства» оправдан тем, что все дополнительные, введенные для удобства формулы адресного языка используют их. Кроме того, основные понятия и средства полностью исчерпывают специфику адресного языка, так как все остальное в языке представляет собой, собственно, комбинации этих понятий и средств с понятиями и средствами обычной математической символики. Перечислим допустимые алгоритмические действия и опишем соответствующие им формулы языка.

Формулы вычисляемого перехода. Запись адресной функции, значениями которой могут быть только метки, использованные в записи алгоритма, может обозначать отдельное алгоритмическое действие, называемое формулой вычисляемого перехода. Такое алгоритмическое

действие состоит в переходе к выполнению строки алгоритма, помеченной меткой, равной значению этой адресной функции, вычисленному при данном адресном отображении.

В частном случае, когда адресная функция постоянна, т. е. не зависит от адресного отображения, формулой вычисляемого перехода может быть метка. При выполнении формулы вычисляемого перехода (в частном случае метки безусловного перехода) адресное отображение не изменяется.

Пример. Пусть некоторые строки алгоритма помечены метками 4 и 6; пусть допустимыми значениями $'a$ являются 3 и 5, тогда в записи алгоритма допустимой будет формула вычисляемого перехода

$$'a + 1.$$

Если значение соответствующей адресной функции не совпадает ни с одной меткой алгоритма, то действие этой формулы вычисляемого перехода считается неопределенным, а соответствующая запись — бессодержательной.

Формулы останова. Употребляется два специальных символа $\mathcal{g}$ и !, называемые, соответственно, формулами относительного и безусловного останова. Употребление символа $\mathcal{g}$ связано с формулами вхождения (см. далее); символ ! означает действие — конец алгоритма. Символ $\mathcal{g}$ также означает конец алгоритма, если ему не предшествовала соответствующая формула вхождения. Формулы останова могут либо составлять отдельные строки алгоритма, либо входить как составные части в другие формулы адресного языка.

Формула относительного перехода. Выражения вида $\updownarrow N$ могут составлять отдельные строки алгоритма и называются формулами относительного перехода. Здесь N — адресная функция, значения которой — положительные или отрицательные целые числа (но не нуль); $\updownarrow$ — специальный символ.

Формула относительного перехода является записью следующего алгоритмического действия: совершить переход к строке, расположенной выше или ниже данной на число строк, равное значению N, вычисленному при заданном адресном отображении (ниже — если N положительно, выше — если N отрицательно).

Формулы засылки. Выражения вида $f_1 \Rightarrow f_2$, (где f_1 и f_2 — адресные функции) могут составлять отдельные строки алгоритма и называются формулами засылки.

Они символизируют следующую алгоритмическую операцию: значение адресной функции f_1 засылается по адресу, равному значению функции f_2. Выполнение формулы засылки изменяет адресное отображение и тем самым значение ряда адресных функций. При этом понимается, что все сказанное при определении операции засылки применяется к значениям функций f_1 и f_2.

Допущение в качестве правой и левой частей операции засылки произвольных адресных функций упрощает запись алгоритмов по сравнению с их записью с помощью основных средств.

Пусть

$$'\alpha = a; \quad 'β = b; \quad 'φ = β; \quad 'ψ = α,$$

тогда $^2φ \Rightarrow 'ψ$ является формулой засылки, после выполнения которой $'\alpha = b$, так как $^2φ = 'β = b$; $'ψ = α$. Следовательно, выполнение этой формулы изменяет значения всех адресных функций, включающих адрес α с любым рангом.

Пусть $'a = 2$; $'b = 0$. Тогда запись

$$'a + 3 \Rightarrow c + 'b$$

означает действие, в результате выполнения которого $'c$ станет равным 5, а запись

$$'a + 3 \Rightarrow 'a$$

будет означать действие, в результате выполнения которого будем иметь

$$'2 = 5.$$

Допустима также запись адресных формул в виде

$$\varnothing \Rightarrow a,$$

где $\varnothing$ — означает символ «пусто» (место для размещения фактических параметров). При этом предполагается, что в алгоритме к моменту обращения к адресной формуле с «пустой» левой частью будет определено, какой символ должен находиться на ее месте.

Для формул засылки, стоящих подряд в одной строке алгоритма и имеющих общие левые части, допускается следующее сокращение записи. Левые части всех, кроме первой, формул опускаются, а разделительный знак запятая или точка с запятой сохраняются.

Например, вместо строки

$$a \Rightarrow b; \quad a \Rightarrow {}'c; \quad a \Rightarrow d + 2$$

можно писать

$$a \Rightarrow b; \Rightarrow {}'c; \Rightarrow d + 2.$$

Формула обмена. Выражения вида $f_1 \leftrightarrows f_2$, (где f_1 и f_2 — адресные функции) могут составлять отдельные строки алгоритма и называются формулами обмена. Они являются записью следующей алгоритмической операции: при заданном адресном отображении вычисляются значения адресных функций f_1 и f_2, которые воспринимаются как адреса, и производится обмен между содержимыми этих адресов. Содержимые остальных адресов остаются неизменными. Так, формула $a \leftrightarrows c$ (где a и c — адреса) означает действие, равносильное действиям, записанным с помощью последовательности трех формул засылки

$$\begin{gathered} {}'a \Rightarrow r \\ {}'c \Rightarrow a \\ {}'r \Rightarrow c. \end{gathered}$$

В общем случае, когда f_1 и f_2 — произвольные адресные функции, это действие равносильно следующим четырем последовательно выполненным формулам засылки:

$$\begin{gathered} a \Rightarrow r \\ {}'c \Rightarrow r_1 \\ {}'a \Rightarrow c \\ {}'r_1 \Rightarrow {}'r. \end{gathered}$$

Здесь r и r_1 рабочие адреса.

На первый взгляд может показаться, что вместо приведенных четырех строк можно написать лишь три строки

$$\begin{gathered} {}'a \Rightarrow c \\ {}'b \Rightarrow a \\ {}'c \Rightarrow b. \end{gathered}$$

В действительности же эти три строки эквивалентны формуле $a \leftrightarrows b$ только в том случае, когда b не зависит от $'a$ (и от всех ${}^n a$, $n \geqslant 1$).

Если же, например, $b = {}'a + 1$ и ${}'a = 3$, $b = {}'a + 1 = 4$, ${}'b = {}'4 = 8$, то по определению формулы обмена после ее выполнения должно быть

$$\begin{gathered} {}'a = 8, \\ {}'4 = 3. \end{gathered}$$

Однако в первом случае получим

$$'a \to c_1$$
$$'b = '('a + 1) = 8 \to c_2$$
$$'a = 3 \to b = 'a + 1 = 4$$
$$'c = 8 \to a,$$

т. е. $'a = 8$; $'4 = 3$. Выполняя три приведенные строки, получим

$$3 \to c$$
$$8 \to a$$
$$3 \to 'a + 1 = 9,$$

т. е. $'a = 8$, а $'9 = 3$, что не соответствует определению формулы обмена.

Список формул. В одной строке алгоритма может записываться несколько формул засылки или обмена. В этом случае между ними ставится точка с запятой или запятая. Первый из этих знаков означает недопустимость перестановки порядка выполнения разделенных им действий, а второй — возможность перестановки или одновременного выполнения этих действий. Подобная строка называется списком формул и означает выполнение всех перечисленных в ней действий с указанными замечаниями относительно порядка их выполнения.

Список формул с переходом. В конце списка формул засылки или обмена, приведенного в одной строке после точки с запятой, может ставиться формула вычисляемого или относительного перехода. Подобная строка называется списком формул с переходом и означает выполнение всех перечисленных в ней действий, последним из которых является переход.

Формулы вхождения и подпрограммы. Укрупнение средств адресного языка в целях лучшей обозримости записей алгоритмов в нем достигается использованием подпрограмм и их формул вхождения. Формулы вхождения применяются для указания того или иного преобразования, которое по тем или иным причинам в данном месте не приводится. Каждая формула вхождения символизирует собой в общей записи алгоритма некоторую многоместную операцию; эта операция может мыслиться как алгоритмически выполнимая (входящая в набор элементарных операций) либо как задаваемая конструктивно некоторым описанием. Само преобразование называется подпрограммой; оно описывается некоторым образом в другом месте

и не обязательно в адресном языке. В описании подпрограммы должны быть указаны:

1) метка, присвоенная данной подпрограмме (название);

2) момент окончания вычислений;

3) упорядоченный список входных и выходных параметров.

К таким параметрам относятся помимо данных об аргументах, размерностях их массивов и т. п., метки тех подпрограмм, использование которых определяется основной программой. Так, например, если подпрограммой является алгоритм вычисления определенного интеграла по некоторой схеме, то помимо границ интегрирования, шага и так далее к входным данным будет относиться указание метки подпрограммы вычисления значений подынтегральной функции.

Подпрограммы в свою очередь могут быть использованы для построения более крупных подпрограмм. Укрупнение подпрограмм в языке за счет ступенчатой структуры их построения ничем не ограничивается.

Подпрограммы, записанные в адресном языке, начинаются строкой, содержащей адресные формулы с «пустыми» левыми частями

$$\varnothing \Rightarrow c_1, \ \ldots, \ \varnothing \Rightarrow c_n .$$

Вместо символов $\varnothing$ в ходе выполнения программы ставятся элементы списка из формулы вхождения. Порядок выполнения этих формул безразличен, но их запись упорядочена и согласована со списком формулы вхождения. Кроме этого, каждая подпрограмма должна содержать, по крайней мере, один символ g, по которому осуществляется выход с нее, т. е. в ходе выполнения алгоритма символ g заменяется формулой β. После выполнения подпрограммы ее запись приобретает исходный вид.

Формулы вхождения подпрограммы имеют вид

$$\Pi\alpha\{a_1, \ \ldots, \ a_n\}\beta.$$

Здесь Π — символ формулы вхождения;

α — формула (в частном случае метка) вычисляемого перехода;

β — формула (в частном случае метка) вычисляемого (относительного) перехода или одна из формул останова;

$a_1, \ldots, a_n$ — список адресных функций.

Эта формула означает:

а) перейти к подпрограмме с меткой α (равной значению α);

б) первые выражения из списка $a_1, \ldots, a_n$ считать упорядоченным списком аргументов подпрограммы;

в) значения остальных из них считать адресами, по которым, выполнив подпрограмму, засылается упорядоченный ряд результатов. Таким образом, число членов этого списка должно быть равно числу входных и выходных параметров данной подпрограммы;

г) перейти к строке алгоритма с меткой β. Причем если β — метка непосредственно следующей строки алгоритма, то в формуле вхождения она может быть опущена.

Формулы вхождения Π в выполнении алгоритмов играют роль открывающих скобок, а закрывающих — символы $\mathcal{J}$.

Символ $\mathcal{J}$ служит формулой останова !, если перед ним не встретилась соответствующая ему формула вхождения.

Программа называется областью действия формулы вхождения.

Предикатные формулы. Предикатными формулами называются выражения вида

$$P\{L\}\,\alpha \downarrow \beta,$$

где P — символ предикатной формулы;

L — некоторое высказывание, т. е. выражение, составленное из букв, числовых констант, символов операций, отношений равенства, неравенства, а также штрих-операций, о котором имеет смысл говорить, что оно истинно или ложно;

$\downarrow$ — разделительный знак;

α и β — верхнее и нижнее значения предикатной формулы, каждое из них может быть одной из описанных ранее видов строк.

Предикатные формулы составляют отдельные строки алгоритма. Соответствующее предикатной формуле алгоритмическое действие состоит в выполнении строки, представляющей верхнее значение предикатной формулы, если соответствующее высказывание истинно, и нижнее значение, если высказывание ложно.

Так, формула

$$P\{1 < 2\}^2\, a \Rightarrow {}'b \downarrow c \leftrightarrows d, {}'c + {}'a \Rightarrow b$$

эквивалентна формуле

$$^2a \Rightarrow {}'b,$$

а формула

$$\boldsymbol{P}\{{}'a = {}'c\}\,{}'f - {}'g \Rightarrow d \downarrow {}'h + 1 \Rightarrow b;\ f$$

при $'a \neq 'c$ эквивалентна выполнению засылки $'h + 1 \Rightarrow b$ и переходу на строку алгоритма, помеченную меткой f.

Если значение L всегда «истина», то вместо предикатной формулы можно записать ее верхнее значение, а если оно всегда «ложь» — ее нижнее значение. При этом все ранее перечисленные допустимые строки являются частным случаем предикатной формулы.

Если в предикатной формуле

$$\boldsymbol{P}\{L\}\,\alpha \downarrow \beta$$

α или β — метка непосредственно следующей за этой формулой строки, то эта метка может опускаться. В этом случае предикатная формула запишется в виде

$$\boldsymbol{P}\{L\} \downarrow \beta$$

или

$$\boldsymbol{P}\{L\}\,\alpha.$$

С помощью только трех видов формул — засылки, останова и предикатной, значениями которой могут быть только метки — можно записать любой алгоритм.

Формулы замены. Формулами замены называются выражения вида

$$\boldsymbol{З}\{a_1 \to c_1,\ \ldots,\ a_n \to c_n\}\,\alpha,\ \beta,\ \gamma,$$

где $\boldsymbol{З}$ — символ формулы;

α, β, γ — формулы вычисляемого или относительного перехода, или просто метки;

$a_1 \to c_1,\ \ldots,\ a_n \to c_n$ — список замен.

Алгоритмическое действие, обозначенное формулой замены, состоит в следующем:

а) при заданном адресном отображении определяются метки — значения формул α и β; строки алгоритма между этими метками определяют область действия формулы замены;

б) выполняются строки алгоритма, составляющие область действия формулы замены, с предварительной за-

меной в них символов c_i соответствующими символами a_i;

в) возможен выход из области действия формулы замены в результате ее исчерпания или вследствие действия какой-либо формулы перехода; в обоих случаях при выходе из области действия восстанавливается в ней первоначальная запись алгоритма;

г) после выполнения формулы замены совершается переход по формуле γ.

Если γ — метка строки, следующей за формулой замены, то она опускается. Формула в этом случае приобретает вид:

$$З\{a_1 \to c_1, \ \ldots, \ a_n \to c_n\}\ \alpha, \ \beta.$$

Формулой замены можно в краткой форме записать участок алгоритма от метки α до метки β, который требуется повторить с другими символами. Заменяться могут как символы адресов, так и символы знаков операций.

Так, например, если в алгоритме вычислялись разности

$$\begin{aligned} \alpha \ \ldots \ & a_1 - b_1 \Rightarrow c_1 \\ & a_2 - b_2 \Rightarrow c_2 \\ & a_3 - b_3 \Rightarrow c_3, \\ \beta \ \ldots & \end{aligned}$$

а требуется выполнить операции

$$\begin{aligned} a_1 + b_1 &\Rightarrow c_1 \\ a_2 + b_2 &\Rightarrow c_2 \\ a_3 + b_3 &\Rightarrow c_3, \end{aligned}$$

то вместо последних можно написать лишь строку

$$З\{- \to +\}\ \alpha, \ \beta.$$

Если область формулы замены составляет одну строку, то метки α и β можно также опускать.

С помощью формулы замены можно записать подпрограммы в другом виде. Вместо строки формул засылки с «пустой» левой частью (см. формулу вхождения) подпрограмма начинается строкой замен вида

$$\varnothing \to c_1, \ \ldots, \ \varnothing \to c_n.$$

Первая строка подпрограммы

$$\varnothing \Rightarrow \varphi, \quad \varnothing \Rightarrow \psi$$

эквивалентна формуле

$$^2a \Rightarrow {}'b,$$

а формула

$$\boldsymbol{P}\{{}'a = {}'c\}\,{}'f - {}'g \Rightarrow d \downarrow {}'h + 1 \Rightarrow b;\ f$$

при $'a \neq 'c$ эквивалентна выполнению засылки $'h + 1 \Rightarrow b$ и переходу на строку алгоритма, помеченную меткой f.

Если значение L всегда «истина», то вместо предикатной формулы можно записать ее верхнее значение, а если оно всегда «ложь» — ее нижнее значение. При этом все ранее перечисленные допустимые строки являются частным случаем предикатной формулы.

Если в предикатной формуле

$$\boldsymbol{P}\{L\}\,\alpha \downarrow \beta$$

α или β — метка непосредственно следующей за этой формулой строки, то эта метка может опускаться. В этом случае предикатная формула запишется в виде

$$\boldsymbol{P}\{L\} \downarrow \beta$$

или

$$\boldsymbol{P}\{L\}\,\alpha.$$

С помощью только трех видов формул — засылки, останова и предикатной, значениями которой могут быть только метки — можно записать любой алгоритм.

Формулы замены. Формулами замены называются выражения вида

$$\boldsymbol{З}\{a_1 \to c_1,\ \ldots,\ a_n \to c_n\}\,\alpha,\ \beta,\ \gamma,$$

где $\boldsymbol{З}$ — символ формулы;

α, β, γ — формулы вычисляемого или относительного перехода, или просто метки;

$a_1 \to c_1,\ \ldots,\ a_n \to c_n$ — список замен.

Алгоритмическое действие, обозначенное формулой замены, состоит в следующем:

а) при заданном адресном отображении определяются метки — значения формул α и β; строки алгоритма между этими метками определяют область действия формулы замены;

б) выполняются строки алгоритма, составляющие область действия формулы замены, с предварительной за-

меной в них символов c_i соответствующими символами a_i;

в) возможен выход из области действия формулы замены в результате ее исчерпания или вследствие действия какой-либо формулы перехода; в обоих случаях при выходе из области действия восстанавливается в ней первоначальная запись алгоритма;

г) после выполнения формулы замены совершается переход по формуле γ.

Если γ — метка строки, следующей за формулой замены, то она опускается. Формула в этом случае приобретает вид:

$$З\{a_1 \to c_1, \ \ldots, \ a_n \to c_n\}\ \alpha,\ \beta.$$

Формулой замены можно в краткой форме записать участок алгоритма от метки α до метки β, который требуется повторить с другими символами. Заменяться могут как символы адресов, так и символы знаков операций.

Так, например, если в алгоритме вычислялись разности

$$\begin{aligned} &\alpha \ \ldots \ a_1 - b_1 \Rightarrow c_1 \\ &\qquad a_2 - b_2 \Rightarrow c_2 \\ &\qquad a_3 - b_3 \Rightarrow c_3, \\ &\beta \ \ldots \end{aligned}$$

а требуется выполнить операции

$$\begin{aligned} a_1 + b_1 &\Rightarrow c_1 \\ a_2 + b_2 &\Rightarrow c_2 \\ a_3 + b_3 &\Rightarrow c_3, \end{aligned}$$

то вместо последних можно написать лишь строку

$$З\{- \to +\}\ \alpha,\ \beta.$$

Если область формулы замены составляет одну строку, то метки α и β можно также опускать.

С помощью формулы замены можно записать подпрограммы в другом виде. Вместо строки формул засылки с «пустой» левой частью (см. формулу вхождения) подпрограмма начинается строкой замен вида

$$\varnothing \to c_1, \ \ldots, \ \varnothing \to c_n.$$

Первая строка подпрограммы

$$\varnothing \Rightarrow \varphi, \quad \varnothing \Rightarrow \psi$$

эквивалентна строке

$$\varnothing \to {}'\varphi, \quad \varnothing \to {}'\varphi.$$

Первая же строка подпрограммы

$$a \to \varphi, \quad b \to \psi, \quad c \to \delta$$

не может быть заменена строкой с адресными формулами; такая подпрограмма не может быть непосредственно записана без формулы замены.

Подпрограммы с первой строкой из формул засылки называют подпрограммами с засылкой. В процессе работы подпрограммы их запись остается неизменной. Подпрограммы с первой строкой, содержащей формулу замены, называют подпрограммами с заменой (с переадресацией). Их запись в процессе работы изменяется, так как для работы подпрограммы одни символы заменяются другими. После окончания работы запись восстанавливается. Допускаются смешанные подпрограммы.

Формулы циклирования. При записи алгоритмов в адресном языке преобразуемые величины могут быть указаны содержащими их адресами некоторых рангов.

Элемент информации называется обозреваемым алгоритмом, если в записи алгоритма содержится его адрес некоторого ранга. Схемами обозревания элементов информации называются алгоритмы, включающие в своей записи адреса некоторого ранга этих элементов.

В общем случае алгоритмы представляют собой схемы обозревания определенных совокупностей элементов исходной информации в определенном порядке. Для установления порядка обозревания используется операция следования.

Естественно, что в общем случае операция следования в множестве элементов исходной информации не может быть непосредственно алгоритмически описана, поскольку информация определяется исходными данными конкретной задачи. Операция следования для элементов исходной информации может устанавливаться с помощью штрих-операции по операции следования в множестве адресов. В задачу программирования входит построение схем обозревания совокупностей элементов с помощью операции следования в множествах их адресов.

Упорядоченное множество запишем в виде

$$\{a,\ C\,\varnothing,\ \boldsymbol{P}\{L\}\}, \tag{1.2}$$

где a — первый элемент;

C — операция следования (на место символа $\varnothing$ можно ставить любой элемент множества);

L — необходимое (но не достаточное) условие принадлежности элемента множеству.

Элементами множества (1.2) являются все те элементы, которые могут быть получены из a путем применения к нему операции C^n $(n = 0, 1, \ldots)$ и удовлетворяют условию L.

Предположим, что в некотором алгоритме требуется выполнить некоторую программу Φ последовательно над всеми элементами множества $\{a,\ C\,\varnothing,\ \boldsymbol{P}\{L\}\}$ и перейти к строке с меткой l.

Такой алгоритм в адресном языке может быть записан в виде

$$\begin{array}{c} a \Rightarrow \pi \\ k\ \ldots\ \boldsymbol{P}\{L\} \downarrow l \\ \Phi\,('\pi) \\ C'\pi \underset{k}{\Rightarrow} \pi. \end{array} \tag{1.3}$$

Для сокращения записи подобных схем вводятся формулы циклирования по адресу π

$$\begin{array}{l} \boldsymbol{Ц}\{a,\ C\,\varnothing,\ \boldsymbol{P}\{L\} \Rightarrow \pi\}\,\alpha,\ l \\ \Phi\,('\pi) \\ \alpha\ \ldots \end{array}$$

или формулы циклирования по параметру π

$$\begin{array}{l} \boldsymbol{Ц}\{a,\ C\,\varnothing,\ \boldsymbol{P}\{L\} \to \pi\}\,\alpha,\ l \\ \Phi\,(\pi) \\ \alpha\ \ldots, \end{array}$$

где $\boldsymbol{Ц}$ — символ формулы;

α — метка строки, следующей за преобразованием Φ.

Строки алгоритма, следующие за формулой до строки с меткой α, называются областью действия формулы циклирования.

Если в область действия формулы циклирования входит другая формула циклирования, замены или вхождения,

то область действия последних включается в область действия первой.

Строка с формулой циклирования может быть помечена и к ней могут быть переходы от других строк алгоритма. Если к формуле циклирования совершен переход по этой метке, то перебор значений π (или $'\pi$) начинается с первого; если же переход осуществлен из области действия в связи с выходом к метке α (конец области действия), перебор значений π (или $'\pi$) продолжается.

Если преобразование Φ содержит переход, в результате которого некоторая часть строк в конце области должна быть опущена, то в конце области действия (перед меткой α) ставится метка с троеточием, отмечающая пустую строку, и указанный переход осуществляется на эту метку. Например, пусть в алгоритме (1.3) $\Phi('\pi)$ имеет вид

$$\Phi_1('\pi)$$
$$\boldsymbol{P}\{L1\}\, k\downarrow$$
$$\Phi_2('\pi).$$

Соответствующая программа с формулой циклирования записывается в виде

$$\boldsymbol{Ц}\{a,\ C\varnothing,\ \boldsymbol{P}(L)\Rightarrow\pi\}\,\alpha,\ l$$
$$\Phi_1('\pi)$$
$$\boldsymbol{P}\{L1\}\, M\downarrow$$
$$\Phi_2('\pi)$$
$$M\ \ldots$$
$$\alpha\ \ldots$$

Таким образом, выход на метку M означает переход к выполнению строк, указанных в области действия формулы циклирования, при следующем значении π или выход из этой области на метку l, если условие L не выполняется.

Для множеств с операцией следования, определяемой соотношением $C\varnothing\Rightarrow\varnothing+b$, употребляется запись

$$\{a\,(b)\,\boldsymbol{P}(L)\}\ \text{или}\ \{a\,(b)\,c\},$$

где c — последний элемент множества.

Формулы циклирования в этом случае имеют вид

$$\boldsymbol{Ц}\{a\,(b)\,\boldsymbol{P}(L)\Rightarrow\pi\}\alpha,\ \beta;\quad \boldsymbol{Ц}\{a\,(b)\,c\Rightarrow\pi\}\,\alpha,\ \beta;$$
$$\boldsymbol{Ц}\{a\,(b)\,\boldsymbol{P}(L)\to\pi\}\,\alpha,\ \beta;\quad \boldsymbol{Ц}\{a\,(b)\,c\to\pi\}\,\alpha,\ \beta.$$

В формуле циклирования возможен одновременный перебор нескольких множеств, например

$$Ц\{a_1(b_1)c_1 \to \pi_1;\quad a_2(b_2)c_2 \Rightarrow \pi_2\}\alpha,\ \beta$$

— циклирование по параметру π_1 и адресу π_2. В этом случае перебор продолжается, пока верны все указанные условия принадлежности, а в формуле может быть указано только одно из них.

Например, запись

$$Ц\{'a_1,\ \varnothing + 1,\ P\{Z_1\} \Rightarrow \alpha;\quad 'b + 2,\ \varnothing + 2 \Rightarrow \beta\}\, m,\ l$$
$$\Phi('\alpha,\ '\beta)$$
$$m\ \ldots$$

эквивалентна записи

$$'a_1 \Rightarrow \alpha;\quad 'b + 2 \Rightarrow \beta$$
$$k\ \ldots\ P\{Z_1\} \downarrow l$$
$$\Phi('\alpha,\ '\beta)$$
$$'\alpha + 1 \Rightarrow \alpha;\ \beta' + 2 \Rightarrow \beta;\ k$$
$$'m\ \ldots$$

Допускается также циклирование по одному адресу в множестве, в котором для начальной группы элементов задана одна операция следования, а для следующей — другая и т. д.:

$$Ц\{a_1,\ C\varnothing,\ P\{Z_1\};\ \ldots;\ a_n,\ C_n\varnothing,\ P\{Z_n\} \Rightarrow \pi\}\alpha,\ l.$$

Элементы множества значений параметра могут задаваться непосредственным перечислением. В этом случае формула циклирования записывается в виде

$$Ц\{a_1;\ a_2;\ \ldots,\ a_n \Rightarrow \pi\}\alpha,\ \beta.$$

Часть алгоритма, состоящую из формулы циклирования по параметру π (по нескольким параметрам) и ее области действия, назовем циклом по этому параметру (по этим параметрам), а область действия формул циклирования — областью действия цикла.

В формуле циклирования допускаются следующие упрощения:

1. Если формула циклирования распространяется только на одну следующую строку, или следующая строка является формулой с указанной областью действия, которая совпадает с областью действия данной формулы

циклирования, то первая метка в формуле циклирования опускается, запятая перед второй меткой при этом сохраняется.

2. Если вторая метка формулы циклирования указывает строку, непосредственно следующую за последней строкой из области действия формулы циклирования, то она опускается вместе со стоящей перед нею запятой.

3. Если обе метки согласно пп. 1 и 2 опускаются, то в формуле опускается ↓ и запятая.

4. Для формулы циклирования по параметру, принимающему значения 1, 2, . . . , n, где n — адресная функция, значения которой — целые положительные числа, вводится сокращенная запись

$$Ц\{(n) \to \pi\}\ \alpha,\ \beta.$$

В этом случае параметр цикла называется счетчиком, работающим по принципу сложения.

Для формулы циклирования по параметру, принимающему значения n, $n-1$, $n-2$, . . . , 2, 1, 0, где n — адресная функция, значениями которой могут быть целые положительные числа, вводится обозначение

$$Ц\{-(n) \to \pi\}\ \alpha,\ \beta.$$

Параметр цикла называется счетчиком, работающим по принципу вычитания.

Формальное определение адресного алгоритма. Запись адресного алгоритма состоит из исходного адресного отображения, конечного числа записанных одна под другой строк и указания множества адресов результативного отображения.

В каждой строке может стоять одна из формул: 1) вычисляемого перехода (метка безусловного перехода); 2) останова; 3) относительного перехода (метка относительного перехода); 4) засылки; 5) обмена; 6) вхождения; 7) предикатная; 8) замены; 9) циклирования или один из списков: 1) формул; 2) формул с переходом.

Любая строка алгоритма может быть помечена.

Начальное адресное отображение задается в виде последовательности равенств

$$'a = b,$$

где a — адрес;

b — содержимое.

Результативное адресное отображение задается перечислением адресов, содержимые которых после окончания работы алгоритма представляют решение задачи.

Порядок выполнения алгоритма определяется следующими правилами:

1. Первой выполняется строка, помеченная специально указанной начальной меткой. Если такого указания нет, то первой выполняется первая строка в записи.

2. Порядок выполнения строк, содержащих списки формул, указан при описании строк.

3. Формулы вычисляемого и относительного переходов, списки формул с переходом, формулы вхождения, замены и циклирования, а также предикатные формулы, значения которых содержат формулы перехода, указывают себе строку-преемницу.

4. После выполнения строки, не указывающей своей преемницы, переходим к следующей в записи строке.

5. Формула безусловного останова! или формула относительного останова $Я$, если ей не предшествует формула вхождения, означают конец работы алгоритма.

Содержательно адресный алгоритм представляет собой некоторую переработку информации о задаче, в результате которой получается ее решение. Эта переработка имеет форму установления и изменения некоторого адресного отображения.

Информация о задаче задается некоторым адресным отображением. Поэтому во многих случаях алгоритм начинается установлением некоторого исходного адресного отображения посредством засылок или же таблицей отображения.

Результатом работы алгоритма является содержимое адресов, указанных в качестве результативного множества.

Описание программы может сопровождаться записанными любым способом пояснениями, которые берутся в специальные квадратные скобки []; требуется, чтобы внутри последних не встречались эти же скобки.

Адресное отображение изменяется в результате выполнения действий засылки или обмена. Формулами циклирования описываются циклические процессы переработки информации, а с помощью формул вхождения отдельные вычисления выделяются в подпрограммы; предикатные формулы могут использоваться для описания разветвляющихся вычислительных процессов или условных перехо-

дов. Нарушение естественного порядка выполнения алгоритма может описываться с помощью формул вычисляемого или относительного перехода. Формула замены может быть использована для записи действия замены одних символов, входящих в запись алгоритма, другими. Формулы останова используются для отметки момента окончания алгоритмического процесса.

Приведем примеры адресных алгоритмов.

Линейный алгоритм. Если строки алгоритма выполняются одна за другой, то алгоритм называется линейным.

Пример 1. *Вычисление выражения*

$$x = \frac{a+b}{d} \cdot c - \frac{a-b}{c} \cdot d.$$

Это можно выполнить по такому адресному алгоритму:

$$\begin{array}{l} k_1 \ldots a + b \to e \\ k_2 \ldots a - b \to f \\ k_3 \ldots c : d \to g \\ \quad 'e \cdot 'g \to e \\ \quad 'f : 'g \to f \\ \quad 'e - 'f \to e, \text{Я} \end{array}$$

После выполнения алгоритма $'e = x$. Можно заранее осуществить исходное адресное отображение, произведя засылки $a \to \alpha$, $b \to \beta$, $c \to \gamma$, $d \to \delta$ или задавая его таблицей:

Адрес	*Содержимое*
α	a
β	b
γ	c
δ	d

Тогда формулы с метками k_1, k_2 k_3 запишутся так

$$\begin{array}{l} k_1 \ldots '\alpha + '\beta \to e \\ k_2 \ldots '\alpha - '\beta \to f \\ k_3 \ldots '\gamma : '\delta \to g \end{array}$$

Объединив в одну строчку формулы, порядок выполнения которых безразличен, получим алгоритм в таком виде:

$$\begin{array}{c} '\alpha + '\beta \to e, \quad '\alpha - '\beta \to f, \quad '\gamma : '\delta \to g \\ 'e \times 'g \to e, \quad 'f : 'g \to f \\ 'e - 'f \to e \\ \text{Я} \end{array}$$

Разбиение алгоритма на отдельные формулы производится произвольно. Можно принимать более малые и более крупные разбие-

Для линейной адресной алгоритм может быть записан и одной строкой. В данном примере

$$\frac{'\alpha + '\beta}{2} \to r_1 \quad \frac{'\alpha - '\beta}{r_1} \times {}'\alpha \to \alpha, \ \Pi.$$

Алгоритм с разветвлением. Если алгоритм зависит от некоторого условия Z, то он может быть реализован с помощью предикатной функции $\boldsymbol{P}\{Z\}$.

Пример 2. *Вычисление функции*

$$y = \begin{cases} x^4 \cdot (x^2 + 1), & \text{если } x < 0; \\ x^4 : (x^2 - 1), & \text{если } x \geqslant 0. \end{cases}$$

Предположим, что $'\alpha = x$; результат $y = '\beta$. При этом

$$\boldsymbol{P}\{'\alpha < 0\}\ ('\alpha)^4 \cdot (('\alpha)^2 + 1) \to \beta \downarrow ('\alpha)^4 : (('\alpha)^2 - 1) \to \beta$$
$$\Pi$$

или

$$\Pi \ \ldots \ ('\alpha)^4 \to r_1, \quad ('\alpha)^2 \to r_2$$
$$\boldsymbol{P}\{'\alpha < 0\}\ 'r_1 \cdot ('r_2 + 1) \to \beta \downarrow 'r_1 : ('r_2 - 1) \to \beta$$
$$\Pi$$

Пример 3. *Нахождение наибольшего общего делителя* $'\alpha$ *и* $'\beta$. Результат получаем по адресу γ. Тогда

$$\begin{array}{l} a \ \ldots \ \boldsymbol{P}\{'\alpha < '\beta\}\ b \downarrow c \\ b \ \ldots \ '\beta - '\alpha \to \beta \\ c \ \ldots \ \boldsymbol{P}\{'\alpha = '\beta\}\ d \downarrow e \\ d \ \ldots \ '\alpha \to \gamma \\ \qquad \Pi \\ e \ \ldots \ '\alpha - '\beta \to \alpha \end{array}$$
$$a.$$

Последний адресный алгоритм можно записать и в таком виде:

$$\begin{array}{l} a \ \ldots \ \boldsymbol{P}\{'\alpha < '\beta\}\ b \\ \qquad \boldsymbol{P}\{'\alpha = '\beta\}\ d \\ \qquad '\alpha - '\beta \to \alpha \\ \qquad\qquad a \\ b \ \ldots \ '\beta - '\alpha \to \beta \\ \qquad\qquad a \\ d \ \ldots \ '\alpha \to \gamma \end{array}$$
$$\Pi.$$

Переставляя формулы, можно сократить одну строку, и тогда алгоритм приобретает вид:

$$\begin{array}{l} b \ \ldots \ '\beta - '\alpha \to \beta \\ B \ \ldots \ \boldsymbol{P}\{'\alpha < '\beta\}\ b \\ \qquad \boldsymbol{P}\{'\alpha = '\beta\}\ d \\ \qquad '\alpha - '\beta \to \alpha \\ \qquad\qquad B \\ d \ \ldots \ '\alpha \to \gamma \end{array}$$
$$\Pi.$$

В связи с тем, что формула, с которой следует начинать выполнение алгоритма, переместилась на второе место, она отмечена меткой *B*.

Наконец, алгоритм может быть записан и в таком виде:

$$B \ldots \boldsymbol{P}\{'\alpha < '\beta\}\ '\beta - '\alpha \to \beta,\ B\downarrow$$
$$\boldsymbol{P}\{'\alpha = '\beta\}\ '\alpha \to \gamma,\quad \mathcal{Я}\downarrow '\alpha - '\beta, \to \alpha,\ B.$$

Алгоритм с использованием формул вхождения. Алгоритм, в котором некоторое вычисление повторяется многократно, может быть реализован с помощью формулы вхождения.

Пример 4. *Вычисление значения функции*

$$f[f(a) + f(b)],$$

где

$$f(x) = \begin{cases} \cos x - 1, & \text{если } x < 0; \\ x^2, & \text{если } x \geqslant 0. \end{cases}$$

Составим предварительно подпрограмму вычисления значения функции f по заданному значению ее аргумента.

Пусть адрес φ используется для хранения аргумента, а по адресу ψ находится адрес, по которому выдается значение функции. Подпрограмма может быть записана в следующем виде:

$$\begin{array}{c} f \ldots \varnothing \to \varphi,\quad \varnothing \to \psi \\ \boldsymbol{P}\{'\varphi < 0\}\ \cos' \psi - 1 \to '\varphi \downarrow ('\varphi)^2 \to '\psi \\ \mathcal{Я}. \end{array}$$

Здесь знаками $\varnothing$ обозначены места для фактических параметров подпрограммы. Первая строка означает, что обращение к данной подпрограмме может быть произведено по формуле вхождения $\Pi f\{\alpha, \beta\}$ с двумя фактическими параметрами α и β, при этом α (аргумент) и β (адрес для помещения результата) ставятся на места символов $\varnothing$.

Напишем такую же подпрограмму с применением формулы замены

$$\begin{array}{c} f \ldots \varnothing \to \varphi,\ \varnothing \to \psi \\ \boldsymbol{P}\{\varphi < 0\}\ \cos \varphi - 1 \to \psi \downarrow \varphi^2 \to \psi \\ \mathcal{Я}. \end{array}$$

Алгоритм для данного примера запишется так:

$$\begin{array}{c} \Pi f\{a,\ \delta\} \\ \Pi f\{b,\ \delta_1\} \\ \Pi f\{'\delta + '\delta_1,\ \delta\} \end{array}$$

Ответ получаем по адресу δ. Алгоритм не зависит от того, к какой из двух подпрограмм, помеченных меткой f, мы обращаемся.

Алгоритм с циклом. Приведем пример записи алгоритма с использованием формулы циклирования.

Пример 5. *Вычисление значений функции $f(x)$, приведенной в примере 4, для ряда значений ее аргумента*

$$x = a_1;\quad a_1 + \Delta a;\ \ldots;\ a_1 + n\Delta a.$$

Результаты разместим по адресам $\alpha,\ \alpha + 1,\ \ldots,\ \alpha + n$.

Предполагая заданной подпрограмму вычисления f и используя формулу цикла, нужный алгоритм можем записать в виде

$$Ц\ (0\,(1)\,n \to \pi_1).!$$
$$Пf\ (a_1 + {}'\pi_1 \times \Delta a,\ \alpha + {}'\pi_1).$$

5.* О РАСШИРЕНИИ ПОНЯТИЯ РАНГА АДРЕСА

Пусть A — множество адресов, для элементов которого установлено отношение различия, т. е. о двух любых адресах можно сказать, равны они или нет. Пусть $\alpha_0 \in A$ — адрес ранга $n\,(n > 0$ — целое) относительно адреса нулевого ранга a, т. е.

$$^{n}\alpha_0 \equiv a.$$

Выпишем в ряд адреса кода a от n-го до 1-го ранга:

$$\alpha_0,\ \alpha_1,\ \ldots,\ \alpha_i,\ \ldots,\ \alpha_{n-1}. \qquad (1.4)$$

Эти адреса связаны соотношением

$$'\alpha_i \equiv \alpha_{i+1}\ (i = 0, 1,\ \ldots,\ n-2),$$

а при $i = n - 1$

$$'\alpha_i \equiv a.$$

Ряд адресов (1.4) назовем последовательностью вложенных в α_0 адресов; α_0 — начальным адресом этой последовательности; код a — ее содержимым. Число членов этой последовательности назовем ее длиной $D(\alpha_0) = n$.

Любой адрес $\alpha_j\ (j = 0, 1,\ \ldots,\ n-1)$ последовательности (1.4) можно рассматривать как начальный адрес некоторой последовательности, выделяющийся из α_j по правилу

$$^{r}\alpha_j \equiv \alpha_{j+r}\ (r = 0, 1,\ \ldots,\ n-j-1).$$

Очевидно, что длина выделяемой из α_j последовательности $D(\alpha_j) = r + 1\ (r = 0, 1\ \ldots,\ n-j-1)$.

Назовем глубиной адреса $\alpha \in A$ величину $\Gamma(\alpha)$, которая равна числу штрих-операций, необходимых для выделения содержимого $a \in A$, т. е. $\Gamma(\alpha) = g$, где $^{g}\alpha \equiv a \in A$.

Легко видеть, что

$$D(\alpha) \leqslant \Gamma(\alpha).$$

Последовательность вложенных адресов, содержимое которой совпадает с одним из членов этой последовательности, т. е. $a \equiv \alpha_i$ (где $i = 0, 1, \ldots, n-1$), назовем замкнутой. Каждый адрес замкнутой последовательности бесконечно глубок.

По определению адресного отображения, если $\alpha \in A$, $\beta \in A$, то из $\alpha \equiv \beta$ следует, что $'\alpha \equiv {}'\beta$.

Если $\Gamma(\alpha) \geqslant n$, $\Gamma(\beta) \geqslant n$, то легко показать, что из $'\alpha \equiv {}'\beta$ вытекает

$$'\alpha \equiv {}'\beta, \quad i = 2, 3, \ldots, n.$$

Можно показать, что если $\Gamma(\alpha_0) \geqslant n$, где $n = i + j$, то

$${}^{i}({}^{j}\alpha_0) \equiv {}^{i+j}\alpha_0 \equiv {}^{j}({}^{i}\alpha_0), \quad (i \geqslant 0;\ j \geqslant 0).$$

Пусть $\beta_0 \in A$ — начальный адрес последовательности вложенных адресов, для членов которой установлена операция следования

$$C\beta_i = \beta_i + k.$$

Здесь k — целое, $j = 0, 1, \ldots, n-2$; $n = D(\beta_j)$. Такие последовательности адресов назовем естественно-вложенными.

Для естественно-вложенных последовательностей адресов выполняются соотношения:

$${}^{r}\beta_j = \beta_j + rk,$$

где

$$r \leqslant \Gamma(\beta_j).$$

При решении информационно-логических задач может возникнуть потребность по данному коду найти его адрес. Эта операция обратна штрих-операции. Назовем ее минус-штрих-операцией[1] и обозначим ${}^{-1}a$.

По данному коду адрес, вообще говоря, определяется неоднозначно — одному коду a может соответствовать целое семейство адресов $\sigma(1, a)$, таких, что для каждого $\alpha \in \sigma(1, a)$ имеем $'\alpha \equiv a$.

Код a назовем адресом минус-первого ранга семейства $\sigma(1, a)$. Адреса семейства σ могут в свою очередь быть адресами минус-первого ранга. Многократное применение минус-штрих-операции приводит к понятию адреса высшего отрицательного ранга.

Из множества A можно выделить подмножество L-массив однозначности, в котором никакие два разные адреса не имеют одинаковых содержимых. На массивах однозначности ${}^{-1}a \equiv \beta$, где $\beta \in L$, т. е. по данному коду адрес определяется вполне однозначно.

---

[1] См. В. П. Сьомик, Про розширення поняття рангу адреси, Обчислювальна математика і техніка, Зб. праць ІК АН УРСР, 1963, № 1.

Условимся в фигурных скобках справа от кода, к которому применяется минус-штрих-операция, указывать массив однозначности. Так запись

$$^{-1}a\,\{z\}$$

означает, что $^{-1}a = \beta$, $\beta \in z$ и ни при каком $\gamma \neq \beta$ и $\gamma \in z$ равенство $'\gamma = a$ невозможно.

Для упорядоченного множества z условимся считать массивом однозначности его отрезок по первый элемент, содержимое которого совпадает с аргументом минус-штрих-операции. Так, например, запись

$$^{-1}371\,\{\alpha + 1\,(1)\,\alpha + n\}$$

означает первый из последовательности адресов

$$\alpha + 1,\ \alpha + 2,\ \ldots,\ \alpha + n,$$

содержащий код 371; запись

$$^{-1}('a \wedge 'b)\,\{\alpha + 1\,(1)\,\alpha + n\}$$

означает первый из адресов той же последовательности, содержащий код, который получается в результате поразрядного логического умножения кодов, содержащихся по адресам a и b.

Результат минус-штрих-операции $^{-1}a\,\{z\}$ будет определен, если в множестве z найдется адрес, содержимое которого совпадает с кодом a. Запись

$$^{-1}a\,\{z\} \Rightarrow b$$

будет означать засылку соответствующего адреса или специального символа $\varnothing$ по адресу b, если код a не содержится ни в одном из адресов множества z.

6. * О ПОЛНОТЕ СРЕДСТВ АДРЕСНОГО ЯЗЫКА

С точки зрения машинной математики особый интерес представляет такое понятие алгоритма, сущность которого раскрывается с помощью представления об устройствах, реализующих алгоритмический процесс.

Способ строго формального определения алгоритмов был предложен в 1936 г. Тьюрингом и назван [17] машиной Тьюринга.

В алгоритме Тьюринга представление о преобразовании информации отождествляется с понятием некоторой аб-

страктной «машины», осуществляющей это преобразование. Описание машины дается ее строго определенной функциональной схемой.

В теории алгоритмов установлено, что с помощью машин Тьюринга могут быть представлены любые алгоритмы, описанные любым другим формальным образом. Поэтому любой мыслимый алгоритм может быть формализован определенной машиной Тьюринга. В связи с этим алгоритмы Тьюринга в настоящее время используются в вопросах теории алгоритмов и математических машин. Поэтому для доказательства универсальности адресного языка [9,24] достаточно показать возможность записи в нем алгоритма произвольной машины Тьюринга.

Машины Тьюринга отличаются от современных ЭВЦМ тем, что в них проведено значительно большее расчленение алгоритмического процесса. Так, в существующих ЦВМ как элементарные операции рассматриваются, например, операция сложения или умножения двух многозначных чисел, выяснения равенства двух чисел и другие; из набора подобных операций составляется алгоритмический процесс (программа). В алгоритмах Тьюринга эти операции расчленяются на цепочки еще более простых операций.

Память машины Тьюринга в отличие от памяти ЭВЦМ изображается бесконечной в оба конца лентой, разделенной на отдельные ячейки. Машина Тьюринга мыслится как устройство, которое в каждом из дискретных моментов времени может находиться в одном из конечного числа своих внутренних состояний и наблюдать лишь одну ячейку памяти. Исходная информация, подлежащая алгоритмическому преобразованию, записывается на ленте в виде слова в некотором конечном алфавите $a_0, a_1, \ldots, a_n$, (содержащем знак «пусто» — a_0).

Символы $\{a_0, \ldots, a_n\}$ составляют внешний алфавит машины.

В любой момент времени в каждой из ячеек хранится один знак, а в пустых — знак пусто.

Каждому отдельному состоянию машины приписывается определенный знак $q_0, q_1, \ldots, q_m$. Среди всех состояний машины выделяется некоторое состояние q_0, после перехода к которому машина останавливается, т. е. прекращается алгоритмический процесс. Находящееся при этом слово на ленте представляет результат алгоритмического процесса.

Работа машины осуществляется отдельными тактами. В каждом такте работы машины в зависимости от состояния q_i, в котором она находится, и от обозреваемого на ленте знака, происходит: 1) замена обозреваемого знака на некоторый знак внешнего алфавита (быть может тот же самый или пустой), 2) сдвиг ленты на одну ячейку вправо, или влево, или сохранение ее положения, 3) переход машины в некоторое из возможных ее состояний (в том числе в состояние, в котором она находится в данном такте). Иначе говоря, можно мыслить себе в машине логический блок, который может находиться в одном из $q_0, q_1, \ldots, q_m$ состояний и перерабатывает информацию согласно таблице действий, которая строится следующим образом.

Число строк таблицы соответствует числу различных возможных состояний логического блока; число столбцов — числу различных букв внешнего алфавита. В каждой графе таблицы записаны три знака: первый знак — буква внешнего алфавита, которая записывается в данном такте в обозреваемую ячейку на ленте; второй знак — один из символов *Л*, *П*, *Н*, соответствующих перемещению ленты на одну ячейку влево, вправо или сохранению ее положения; третий знак — один из символов, соответствующих возможным состояниям машины. Символы *Л*, *П*, *Н* и $q_0, \ldots, q_m$ составляют внутренний алфавит машины.

По заданному исходному состоянию и обозреваемому знаку логический блок выдает приказ из трех символов соответствующей клетки таблицы, в результате выполнения которого совершается один такт алгоритмического процесса. В зависимости от начальной информации возможны два исхода:

1) после конечного числа тактов машина останавливается (состояние q_0 достижимо);

2) машина никогда не останавливается (состояние q_0 недостижимо).

В первом случае считаем, что машина применима к заданному слову, во втором — не применима.

Пример. *Построение машины Тьюринга для сложения целых чисел.* В качестве внешнего алфавита возьмем {I, Λ, *}. Букву I используем для кодирования целых чисел. В этом случае, например, число единица запишется как I, число пять как IIIII и т. п; * используем как разделительный знак для записи слагаемых на ленту; Λ — означает пустой знак.

Работа машины Тьюринга, осуществляющей сложение целых чисел, может быть сведена к сдвигу всех черточек, изображающих первое слагаемое, на одну ячейку вправо. При этом разделительный знак уничтожится и полученное на ленте слово будет результатом.

Для реализации такого процесса достаточно двух различных состояний машины q_1 и q_2 и останова q_0.

Предположим, что в начальный момент машина находится в состоянии q_1 и обозревает самый левый знак исходного слова.

В начальном состоянии машина находится только в первом такте своей работы. При этом:

1) при обозревании знака $*$ она стирает его и переходит в конечное состояние q_0, т. е. останавливается. Это произойдет в том случае, когда первое слагаемое равно нулю;

2) при обозревании знака $\wedge$ машина останавливается. Это произойдет, если оба слагаемых равны нулю — входное слово пустое;

3) при обозревании знака I машина стирает последний, сдвигается вправо и переходит в состояние q_2 (в этом случае первое слагаемое не равно нулю).

Пусть в состоянии q_2 машина при обозревании знака I сдвигается вправо, пока не найдет один из знаков $*$ или $\wedge$, и затем останавливается, причем знак $*$ предварительно заменяется на I.

Функциональная таблица машины имеет вид табл. 1.

Для сокращения записи функциональной табл. 1 некоторые из символов могут быть опущены, и табл. 1 запишется в виде табл. 2.

Таблица 1

q	a		
	I	$\wedge$	$*$
q_0	IHq_0	$\wedge Hq_0$	$*Hq_0$
q_1	$\wedge Pq_2$	$\wedge Hq_0$	$\wedge Hq_0$
q_2	IPq_2	$\wedge Hq_0$	IHq_0

Таблица 2

q	a		
	I	$\wedge$	$*$
q_1	$\wedge Pq_2$	q_0	$\wedge q_0$
q_2	P	q_0	Iq_0

Пусть, например, необходимо сложить числа: 1) 3 и 2; 2) 0 и 2; 3) 4 и 0.

Выпишем в столбцы все этапы преобразования

1) $\dot{I}II * II$
$\wedge\dot{I}I * II$
$\wedge I\dot{I} * II$
$\wedge II\dot{*} II$
$\wedge II\ \dot{I}\ II$

2) $\dot{*}II$
$\wedge\dot{I}I$

3) $\dot{I}III *$
$\wedge\dot{I}II*$
$\wedge I\dot{I}I*$
$\wedge II\dot{I}*$
$\wedge III\dot{*}$
$\wedge IIII$

Точкой сверху отмечаем обозреваемый на данном такте символ.

Приведем другой вариант машины, реализующей этот же алгоритм. Кодирование информации осуществим так же, как и в предыдущем случае. Однако процесс сложения чисел сведем к приписыванию первого (левого) слагаемого ко второму справа от него и к стиранию разделительного знака. В этом случае понадобится машина с тремя рабочими состояниями q_1, q_2, q_3, из которых q_1 — начальное; считаем, как и прежде, что к началу работы машины обозревается первый слева знак входного слова.

Данный алгоритм описывается табл.3. Рекомендуем читателю проследить за работой машины на частных примерах.

Таблица 3

q	a		
	I	$\wedge$	$*$
q_1	$\wedge \Pi q_3$	Π	$\wedge q_0$
q_2	Π	Πq_1	$\wedge$
q_3	Π	$\mathrm{I} q_2$	Π

Рассмотрение даже этого простого примера показывает всю сложность построения и работы машины Тьюринга.

Перейдем к адресному описанию машины Тьюринга. Память машины Тьюринга состоит из множества упорядоченных адресов $\{\alpha\}$

$$\ldots, \alpha - 1, \alpha, \alpha + 1, \ldots$$

Записи исходной информации в ячейках ленты в адресном языке соответствует распределение знаков исходного алфавита по адресам $\{\alpha\}$. Головке машины Тьюринга, обозревающей в каждый данный момент одну из ячеек и воспринимающей содержащийся в ней знак алфавита, в адресном алгоритме поставим в соответствие некоторый фиксатор — адрес φ, содержимым которого является адрес обозреваемой ячейки из $\{\alpha\}$; тогда $^{2}\varphi$ является обозреваемым знаком алфавита. Сдвигу головки будет соответствовать изменение содержимого адреса φ : $'\varphi + 1 \Rightarrow \varphi$ при сдвиге вправо; $'\varphi - 1 \Rightarrow \varphi$ — в случае сдвига влево; $'\varphi + 0 \Rightarrow \varphi$ — в случае отсутствия сдвига; вообще $'\varphi + \delta \Rightarrow \varphi$, где $\delta = +1, -1, 0$ соответственно. Символы состояний $\{q\}$, в которых машина может находиться, будем помещать по адресу ψ.

В соответствии с требованием линейности записи адресного алгоритма действия, указанные в клетках функциональной таблицы машины Тьюринга, запишем одно под другим в строки адресного алгоритма, поставив перед ними соответствующие метки

$$a_i \bigcirc q_j \ldots,$$

состоящие из трех последовательно выписанных символов: буквы внешнего алфавита, специального знака ○ и символа состояния.

Поиск нужной млетки, т. е. строки в адресном алгоритме, с которой начинаются нужные действия, производится алгоритмически по строке адресного алгоритма, состоящей из формулы безусловного перехода

$$^{2}\varphi \circ {}'\psi.$$

Эту строку отметим меткой B.

Действиям, записанным в клетках таблицы машины Тьюринга, в адресном алгоритме будут соответствовать строки

$$\begin{gathered} a_i \circ q_l \ \ldots \ a_{ij} \Rightarrow {}'\varphi;\ q_{ij} \Rightarrow \psi \\ {}'\varphi + \hat{0} \Rightarrow \varphi \\ B. \end{gathered}$$

Любая из формул может быть опущена, если она не изменяет адресного отображения.

В отдельных клетках таблицы может находиться формула останова (так называемые конечные состояния). Для таких клеток вместо B будет !.

Так, приведенный алгоритм тьюринговского типа с алфавитом {|,∧,∗} и состояниями q_1, q_2, q_0 для сложения целых положительных чисел (табл. 2) можно записать в адресном виде:

$$\begin{gathered} B \ \ldots \ {}^{2}\varphi \circ {}'\psi \\ | \circ q_1 \ \ldots \ \wedge \Rightarrow {}'\varphi;\ q_2 \Rightarrow \psi \\ {}'\varphi + | \Rightarrow \varphi \\ B. \\ | \circ q_2 \ \ldots \ {}'\varphi + | \Rightarrow \varphi \\ B \\ \wedge \circ q_1 \ \ldots \quad ! \\ \wedge \circ q_2 \ \ldots \quad ! \\ * \circ q_1 \ \ldots \quad \wedge \Rightarrow {}'\varphi;! \\ * \circ q_2 \ \ldots \quad | \Rightarrow {}'\varphi;! \end{gathered}$$

Для описания произвольной машины Тьюринга, т. е. для построения любого мыслимого алгоритма, понадобились только немногие из средств адресного языка:

1) засылка по адресу $a \Rightarrow b$;

2) засылка по адресу, являющемуся содержимым другого адреса, $a \Rightarrow {}'b$;

3) безусловный переход по формуле, содержащей адрес второго ранга, $^2a \subset {}'b$;

4) сдвиг фиксатора ${}'a + \delta \Rightarrow a$.

Отказавшись от арифметизации упорядоченности адресов $\{\alpha\}$, можно для сдвига фиксатора ограничиться некоторыми символами, например $П'\varphi \Rightarrow \varphi$ — сдвиг направо, $Л'\varphi \Rightarrow \varphi$ — сдвиг налево.

Описав машину Тьюринга в адресном языке каким-либо другим способом, можно выделить иное множество элементарных средств адресного языка, составляющих полный набор алгоритмических операций.

Пользуясь же адресным языком свободно, т. е. не стремясь к использованию только минимума его средств, можно построить алгоритм машины Тьюринга более простым способом. Оставим те же обозначения, что и в предыдущем случае, но информацию о функциональной таблице машины разместим по адресам. В каждой клетке такой информацией служит элемент a_{ij}, который нужно вписать в обозреваемую ячейку; новое состояние машины (конечное состояние обозначим q_0), и, наконец, величина δ_{ij}, равная 1, —1, 0. Условимся также считать знаки алфавита $a_1, a_2, \ldots, a_n$ и знаки состояний $q_1, q_2, \ldots, q_m$ рядами натуральных чисел 1, 2, ..., n и 1, 2, ..., m. Здесь речь идет только о перекодировке, а не об изменении алгоритма.

Разместим элементы информации о таблице построчно в массив адресов $\gamma + 1, \gamma + 2, \ldots$ Тогда элементы первой клетки попадут в адреса ${}'(\gamma + 1) = a_{11}$; ${}'(\gamma + 2) = q_{11}$; ${}'(\gamma + 3) = \delta_{11}$; элементы второй — в адреса ${}'(\gamma + 4) = a_{12}$; ${}'(\gamma + 5) = q_{12}$; ${}'(\gamma + 6) = \delta_{12}$; элементы клетки, стоящей на пересечении i-й строки и j-го столбца, попадут в адреса

$$
\begin{aligned}
&{}'(\gamma + 3m(i-1) + 3(j-1) + 1) = a_{ij};\\
&{}'(\gamma + 3m(i-1) + 3(j-1) + 2) = q_{ij};\\
&{}'(\gamma + 3m(i-1) + 3(j-1) + 3) = \delta_{ij}.
\end{aligned}
$$

Алгоритм любой машины может быть теперь записан так:

$$
\begin{aligned}
&B \ldots \gamma + 3'M({}^2\varphi - 1) + 3({}'\psi - 1) \Rightarrow r\\
&\boldsymbol{P}\{{}'({}'r + 2) = q_0\}!\\
&{}'({}'r + 1) \Rightarrow {}'\varphi; \quad {}'({}'r + 2) \Rightarrow \psi\\
&{}'\varphi + {}'({}'r + 3) \Rightarrow \varphi\\
&\qquad B.
\end{aligned}
$$

Здесь ${}'M = m$.

Следует особо подчеркнуть одно из основных свойств адресного языка, заключающееся в удобстве написания в нем произвольных обобщенных алгоритмов, т. е. алгоритмов, решающих неограниченно широкий класс задач. При этом вся информация не только о конкретных условиях задачи, но и о классе задач может быть отнесена к исходному адресному отображению, т. е. включена в исходную информацию о задаче. Алгоритм же может быть записан общий, не зависимый от решаемой задачи. Так, приведенный адресный алгоритм является адресной записью любого алгоритма.

Конечно, на практике главная трудность решения конкретной задачи переносится при этом на заполнение начального массива памяти, но это не умаляет значения сделанного замечания. Адресный язык допускает перенесение любого объема исходной информации в адресное отображение с целью упрощения и обобщения записи алгоритма. Приведенный пример показывает, что единственной информацией, которая не может быть перенесена в область адресного отображения и всегда необходима для записи адресного алгоритма, является информация о необходимости записать какой-то алгоритм.

Это свойство адресного языка позволяет, при пользовании им, абстрагироваться от конкретных особенностей алгоритмов и исследовать общие проблемы теории алгоритмов. В программировании, используя адресный язык, можно решать конкретные задачи, в частности связанные с конкретными вычислительными машинами.

* 7. АДРЕСНЫЙ ЯЗЫК И ПРИНЦИП АДРЕСНОСТИ В АЛГОРИТМИЧЕСКИХ ЯЗЫКАХ

К алгоритмическим языкам предъявляется ряд требований, среди которых наряду с алгоритмической полнотой существенное значение имеет простота достижения общности записи алгоритмов в этих языках.

При построении алгоритмического языка в качестве исходного часто берется алгебраический язык формул. Для указания хода преобразования информации вводятся специальные операторы. Однако в алгоритмических языках в отличие от исходного алгебраического языка уточняется понятие памяти, называемое принципом адресности.

В настоящее время, в связи с развитием методов автоматизации программирования, появился ряд алгоритмических языков. Эти языки в различной степени не зависят от конкретных языков ЭВЦМ, отличаются способами записи, набором операций и операторов и имеют различную направленность: для описания вычислительных процессов, преобразования буквенной информации, описания производственно-экономических задач и т. п.

Несмотря на все разнообразие создающихся для программирования языков, по степени использования принципа адресности их можно разделить на следующие три группы.

Для всякого алгоритма существует некоторая совокупность возможных исходных данных I — множество конструктивных объектов, к которым имеет смысл применять данный алгоритм. Для построения заданного алгоритма используется некоторая совокупность конструктивных объектов K — множество структурных элементов алгоритма.

Эти множества могут пересекаться. При этом чем больше используются элементы исходной информации в качестве структурных элементов алгоритма, тем, вообще говоря, менее общей будет запись алгоритма.

Из множества K выделим следующие подмножества:

1. Непустое множество $A \subset K$, называемое множеством адресов. Множество A содержит некоторые упорядоченные подмножества A_i ($A_i \subset A$).

2. Непустое множество O, называемое множеством операций. Оно состоит из двух подмножеств O_I и O_K. Множество O_K представляет набор алгоритмических операций, не зависящих от природы исходной информации и удовлетворяющих свойству алгоритмической полноты. Элементы множества O_I вводятся для удобства записи алгоритмов; тот или иной их набор определяется природой исходной информации.

3. Некоторое непустое множество B — совокупность вспомогательных элементов, как-то: метки, разделители и т. п.

Обычно в начале работы алгоритма устанавливается соответствие между множеством I и некоторым подмножеством множества A. При применении к элементам множества A как к аргументам специальной операции из O_K, называемой штрих-операцией, получим множество значений C, называемое множеством содержимых.

Отображение множества A на C называется адресным отображением.

В начале работы алгоритма, если $i \in I$, существует такое $c \in C$, что $i = c$. В процессе работы алгоритма адресное отображение изменяется и по окончании работы, если $r \in R$ (R — результативное множество), существует такое $c_r \in C$, что $c_r = r$. Задачей всякого алгоритма является получение для заданного множества I результативного множества R.

Чтобы не снижать общность алгоритмов присутствием элементов исходной информации в их записи, разделяем запись алгоритмов на две части — статическую и динамическую.

Статическая часть S служит для установления исходного адресного отображения; она описывается на некотором языке соотношений, в котором явно представлены элементы исходной информации.

Динамическая часть D описывается на языке процессов. Исходную информацию здесь представляют элементы множества C ($C \subset K$). Динамическая часть алгоритма при переходе на язык машины представляется в виде программы.

Как структурные элементы алгоритмов элементы множества C могут играть различную роль в записи алгоритмов, определяемую степенью использования в языке принципа адресности. Различаем три ступени языка.

1. Множество C представляет только исходную информацию. Поэтому имеет смысл штрих-операция только первого ранга, и в множестве O_K необходима операция переадресации (замены). С другой стороны, чем разнообразнее природа исходной информации, тем естественнее вводить более широкие наборы операций O_I. В статической части устанавливается соответствие лишь между исходной информацией и множеством A; удельный вес ее незначителен и поэтому деление на S и D часто вообще не делается. Общность, достигаемая на этом уровне, соответствует общности, достигаемой в алгебре при введении обозначения чисел буквами.

2. Содержимые могут представлять собой адреса. При этом в статической части устанавливается соответствие не только между I и A, но и между некоторым подмножеством $A_C \subset A$ и всем множеством A, т. е. C и A пересекаются и их общей частью является A_C. Во множестве

O_K может быть введена штрих-операция второго ранга. Набор операций O_1 определяется, как и на I ступени, природой исходной информации. Вообще язык I ступени является подъязыком II ступени. Язык II ступени отличается от языка I ступени высокой общностью записи в нем алгоритмов; при этом важную роль играет статическая часть, за счет которой можно уменьшать динамическую часть.

3. Содержимые представляют и элементы множества O. На этой ступени допускается пересечение множеств O и C. При этом из записи алгоритмов можно вообще исключить элементы множества O_1, относя их к исходному адресному отображению. Тогда динамическая часть становится не зависящей от исходной информации. Запись алгоритмов на этой ступени максимально общая. Более того, ее можно сделать единой для всех мыслимых алгоритмов, записываемых на языке II ступени, и тем более — I ступени. Если же в записи алгоритмов допустить множество O_1, то язык II ступени будет подъязыком III ступени, т. е. получается достаточно гибкий универсальный язык.

Первому уровню соответствуют языки, не имеющие операций по адресам высших рангов, но имеющие операцию замены или специальные операции над индексами, например АЛГОЛ [15] и адресный язык, не использующий операции по адресам высших рангов.

Для расширения адресного языка II ступени до III ступени достаточно переопределить только понятие основной структурной части формул — адресной функции. Остановимся на определении адресной функции в строго скобочной записи[1], так как при неявном задании операций говорить о старшинстве операций неудобно.

Определим теперь адресную функцию так, чтобы в множество новых функций входили функции языка II ступени.

Рассмотрим две вспомогательные компоненты (одноместную операционную компоненту $< k_1 >$ и двухместную операционную компоненту $< k_2 >$), а также используем определение компоненты $< k_{II} >$ языка II ступени.

---

[1] См. [1] или стр. 147

Тогда

$< k_1 > : : = O_1 \mid <$ штрих-операция положительного ранга $> < k_{II} >$

$< k_2 > : : = O_2 \mid <$ штрих-операция положительного ранга $> < k_{II} >$.

Заменяя в определении $< k_{II} >$ символы O_1 на $< k_1 >$, а O_2 на $< k_2 >$, получим определение компоненты языка III ступени

$$< k_{III} > : : = a \mid < k_1 > < k_{III} > \mid (< k_{III} > < k_2 > < k_{III} >),$$

и соответственно адресной функции F языка III ступени

$$< F > : : = a \mid < k_1 > < k_{III} > \mid < k_{III} > < k_2 > < k_{III} >.$$

Определим также компоненту отношения $< k_3 >$ следующим образом:

$$< k_3 > : : = O_3 \mid <$$ штрих-операция $> < k_{II} >.$$

Логическое условие предикатной формулы языка III ступени определится так:

$$< L > : : = < F > < k_3 > < F >.$$

В полученном таким образом языке В. П. Семиком построен универсальный алгоритм, преобразующий адресные алгоритмы к общему виду (см. § 5 этой главы), при котором они отличаются только статической частью, а динамическая за счет известного расширения статической становится одинаковой для всех. Идею построения этого алгоритма поясним на примере адресных функций.

Подобными компонентами на соответствующих ступенях считаются:

1) полностью совпадающие;

2) отличающиеся только адресами, например

$$^2a + (' b - {}^2d);$$
$$^2c + (' e - {}^2f);$$

3) совпадающие по форме, например

$$^3a + (' b - {}^2d);$$
$$^2c : (^2e - ' d).$$

Вначале осуществляется экономия одинаковых в
вений первой категории. Далее путем введения адр
вленных рангов и адресов для символов операций подо
вождения второй и третьей категории превраща
в одинаковые и экономятся как и на I ступени. При
статическая часть алгоритма пополняется за счет ви
вых адресов вызовных рангов, а динамическая соотве
венно сокращается до приобретения некоторой кан
ческой формы.

ГЛАВА II

ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ЦИФРОВЫЕ МАШИНЫ (ЭВЦМ)

1. БЛОК-СХЕМА

Современные вычислительные машины и устройства в зависимости от способа представления в них величин разделяют на два больших класса — цифровые (дискретного счета) машины и непрерывного действия, или аналоговые.

В машинах непрерывного действия числа представляются в виде тех или иных физических величин — длин отрезков, углов поворота, электрического напряжения и др. Каждая из этих величин может принимать (в некоторых пределах) произвольные значения и изменяться на сколь угодно малые величины. Точность представления чисел в этих машинах ограничивается точностью измерения соответствующих величин и практически равна 3—4 десятичным знакам.

К аналоговым относятся такие приборы, как логарифмическая линейка, электронные дифференциальные анализаторы типа МПТ-9 и др.

В отличие от аналоговых машин в ЭВЦМ числа представляются, как и при записи их на бумаге, в цифровом виде. С этой целью каждая цифра числа кодируется в избранной системе счисления. Для кодирования используются элементы различной физической природы, имеющие необходимое количество устойчивых состояний. Каждому из этих состояний приписывается одна из цифр принятой системы счисления. Так, с помощью механической системы, например колеса Однера или набора косточек, осуществляется запись чисел в десятичной системе счисления в простейших цифровых приборах — арифмометрах или счетах.

Это дает практически неограниченную точность. Поэтому универсальные ЭВЦМ предназначаются для решения различных вычислительных и логических задач с большим объемом вычислений и высокой точностью.

Таким образом, общие принципы построения универсальных ЭВЦМ до некоторой степени аналогичны принципам конструирования ранее известных цифровых машин, например электрических клавишных машин и др.

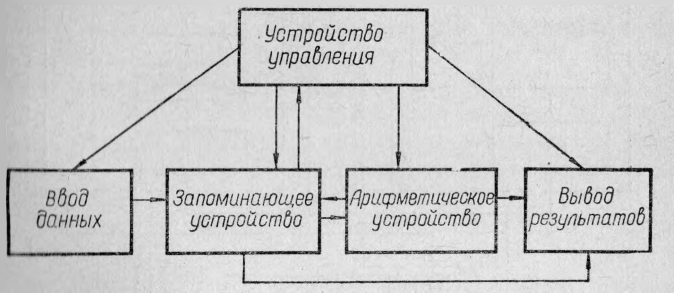


Рис. 1.

Однако принципиально новым в ЭВЦМ, помимо использования современных средств электроники, является применение схем, обеспечивающих запоминание результатов вычислений, полную автоматизацию всего вычислительного процесса и исключительно большую скорость его.

Алгоритм решения задачи для реализации на ЭВЦМ должен быть расчленен на последовательность элементарных операций, выполняемых машиной.

Таким образом, универсальная ЭВЦМ состоит из имеющих специальное назначение блоков:

1) устройства ввода данных;
2) запоминающего устройства (ЗУ);
3) арифметического устройства (АУ);
4) устройства (автоматического и ручного) управления (УУ);
5) устройства вывода результатов.

Блок-схема ЭВЦМ приведена на рис. 1; стрелки указывают связь между отдельными устройствами, заключающуюся в возможности передачи закодированной информации (хранящейся или вырабатываемой в блоке) из этого устройства в другое.

Устройство ввода предназначается для передачи в ЗУ необходимой для решения задачи информации.

Чаще всего работа этого устройства осуществляется автоматически с помощью перфорированных карт или лент. В зависимости от быстродействия машины для ускорения ввода применяются читающие автоматы, магнитные ленты, на которых данные записываются в виде

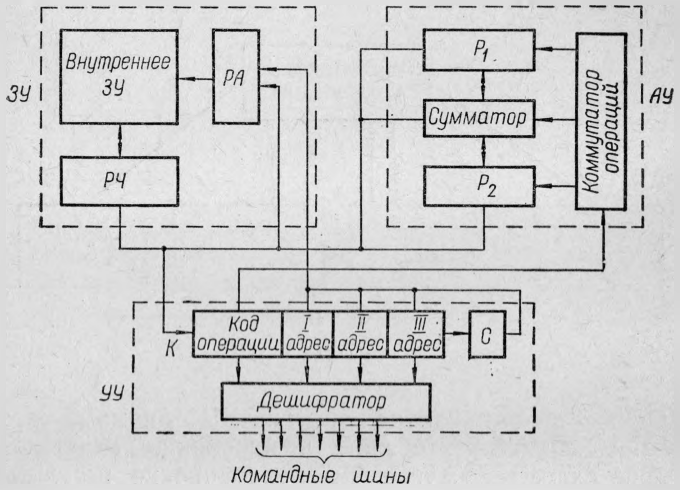


Рис. 2.

магнитных импульсов, и др. Схема связей между отдельными устройствами *ЗУ* приведена на рис. 2.

Устройства *ЗУ*, *АУ*, *УУ* предназначены для непосредственного выполнения вычислительных операций.

ЗУ служит для приема, хранения и выдачи кодов программы и результатов промежуточных и окончательных вычислений.

ЗУ помимо ячеек (чаще всего состоящих из одинакового числа разрядов) имеет вспомогательные регистры — регистр адреса (*РА*) и регистр числа (*РЧ*), — которые используются для передачи кодов из *ЗУ* в другие устройства машины и для приема кодов в *ЗУ*. Для выборки числа, хранящегося по адресу a, последний поступает в *РА*, с помощью избирательной схемы включается ячейка с данным адресом, и ее содержимое считывается в *РЧ* и далее в другие устройства.

Арифметическое устройство представляет собой объединение электронных вычислительных схем, с колоссальной скоростью выполняющих различные операции над поступающими в него кодами. При этом каждой машине присущ свой набор операций, выполняемых *АУ* над кодами. Обычно *АУ* состоит из двух приемных регистров, сумматора и дешифратора операций; для выполнения каждой отдельной операции *АУ* специальным образом настраивается с помощью дешифратора *АУ*.

УУ управляет последовательностью операций в соответствии с заданной программой.

Обычно в *УУ* входит регистр команд *(К)*, подразделяемый в зависимости от адресности машины на несколько адресных подрегистров, счетчик команд *(С)* и дешифратор *(Д)*. Действие *УУ* определяется в каждый отрезок времени содержимым его регистра *К*.

Программа задачи состоит из отдельных слов — команд, которые в определенном порядке поступают на регистр *К* и тем самым определяют работу *УУ* и всей машины в целом. С помощью дешифратора осуществляется выдача управляющих импульсов на командные шины, реализующие связи со всеми блоками машины. Счетчик команд *С* служит для выборки команд из ячеек *ЗУ*; он хранит адрес команды, т. е. номер ячейки *ЗУ*, из которой извлекается команда для выполнения.

Устройство вывода предназначается для выдачи из машины окончательных результатов счета; чаще всего оно реализуется в виде печатающего устройства, с помощью которого искомые результаты наносятся в определенной последовательности, например, на ленты.

2. КОДИРОВАНИЕ ИНФОРМАЦИИ

Существенным и принципиальным преимуществом цифрового способа записи чисел помимо высокой точности является то, что с помощью цифр может быть представлена любая другая информация, например буквенная. Одним из таких способов является азбука Морзе.

Для цифрового кодирования букв некоторого (конечного) алфавита можно условиться для каждой буквы отводить одинаковое число разрядов. Например, для кодирования 40-буквенного алфавита, состоящего из букв

a_1, a_2, ..., a_{40}, можно поступить следующим образом. Выбираем такое минимальное число n, чтобы $2^n \geqslant 40$. Для этого достаточно взять $n = 6$. Теперь пусть букве a_1 ставится в соответствие двоичное число 000001, букве a_2 — 000010, т. е. i-й букве исходного алфавита a_i ставится в соответствие целое двоичное число, равное ее номеру; если для записи числа потребуется меньше 6 разрядов, то впереди приписывается соответствующее число нулей.

Таким образом, в зависимости от кодирования (принятого соглашения) цифровые коды могут рассматриваться как числа или символы любой другой природы.

Всякая математическая машина представляет собой устройство, способное выполнять некоторый фиксированный набор операций (преобразований записей) и выдавать или сохранять в себе сведения о результатах этих операций.

Наибольший интерес представляют машины, допускающие выполнение произвольных последовательностей операций, — универсальные вычислительные машины. Переход от решения одной задачи к решению другой в универсальных машинах («Урал», «Стрела», БЭСМ, «Киев» и др.) производится без каких-либо изменений в схемах их коммутаций в отличие от специализированных машин.

Практически почти любая задача, доступная методам численного анализа, может быть решена на универсальной машине. Кроме того, соответствующим кодированием (шифровкой) исходной информации и сведением правил ее преобразования к последовательностям операций, выполнимых в данной машине, на универсальных вычислительных машинах решаются такие весьма далекие от вычислений задачи, как машинный перевод с одного языка на другой, тождественные преобразования формул (алгебраических и алгебры логики), участие машины в играх (в качестве одного из партнеров), поиск кратчайшего пути в лабиринте, поиск «слова» в «словаре» и др. Однако все эти применения машин становятся возможными лишь после надлежащего уточнения, а зачастую и составления алгоритма решения той или иной задачи.

Выбор способа кодирования информации определяет в известной мере формулировку алгоритма, что легко можно себе уяснить хотя бы на примере алгоритма сложения чисел. Действительно, формальные правила сложения чи-

сел в десятичной записи (системе счисления) и, например, в двоичной, содержательно совпадающие между собой, получаются различными. Это относится и к решению задач на машинах.

Поскольку при подготовке задач с числовой информацией пользуются десятичной системой счисления, а машины оперируют с двоичными числами, для ввода чисел в память машин, для возможности контроля работы машин, для вывода чисел из памяти на выводное устройство приходится переводить числа из одной системы в другую. Для упрощения правил перевода, а также в связи с тем, что записи чисел в двоичной системе счисления слишком однообразны и длинны, удобно пользоваться наряду с десятичной и двоичной системами счисления некоторыми другими. Такими широко распространенными в машинах являются восьмеричная и двоично-десятичная системы.

Восьмеричная и шестнадцатеричная системы удобны тем, что их основания являются степенями двойки. Так, $8 = 2^3$, $16 = 2^4$. В связи с этим одному разряду записи числа в восьмеричной (шестнадцатеричной) системе соответствуют три (четыре) разряда его записи в двоичной системе. Так, число десятичное 151 запишется в двоичной системе как 10010111. Разбив его справа налево на тройки (триады) (10.010.111) и записав каждую тройку одной цифрой, получим запись того же числа в восьмеричной системе 227.

Разбив на четверки (тетрады) (1001.0111), получим шестнадцатеричную запись 97.

Для шестнадцатеричной системы счисления приходится вводить дополнительные цифры: $\overline{0}$-10, $\overline{1}$-11, $\overline{2}$-12, $\overline{3}$-13, $\overline{4}$-14, $\overline{5}$-15.

В двоично-десятичной системе каждая десятичная цифра числа записывается двоичным кодом в четверке разрядов (тетраде). Соответствие четверок двоичных цифр и цифр десятичной записи то же, что и для шестнадцатеричных, но двоичные коды 1010, 1011, 1100, 1101, 1110, 1111 не имеют смысла и в двоично-десятичной записи не употребляются.

Числа в ЭВЦМ кодируются чаще всего в двоичной системе счисления. Таким образом, каждый разряд запоминающей ячейки *ЗУ* содержит одно из чисел: 0 или 1.

Один разряд ячейки *ЗУ* используется для кодировки знака:

$$\text{«}+\text{»} = 0$$
$$\text{«}-\text{»} = 1.$$

Кодирование чисел осуществляется одним из двух способов в зависимости от особенностей машины (машины с фиксированной и с плавающей запятой).

В машинах с фиксированной запятой применяется естественная форма записи чисел: число представляется последовательностью цифр, разделенной на целую и дробную части. В большинстве машин запятая фиксирована непосредственно после знакового разряда. В машинах с плавающей запятой каждое число a представляется в виде

$$a = 2^p \cdot x,$$

где p — двоичный порядок числа со своим знаком;

$x = a_0, a_1 a_2 \ldots a_n$ — цифровая часть числа, называемая мантиссой (a_0 — знак мантиссы);

n — число разрядов ячейки ЗУ, отведенных для цифровой части;

$a_i = 0$ или 1 $(i = 1, \ldots, n)$. Цифровая часть числа обычно представляется в нормализованном виде, т. е. $a_1 = 1$.

Таким образом, в машинах с фиксированной запятой каждый разряд используется для кодировки цифр числа, а в машинах с плавающей запятой выделяется несколько разрядов для кодирования порядка числа и его знака, остальные разряды используются для цифровой части числа.

Таким образом, в машине с фиксированной запятой коду

$$a_0 a_1 \ldots a_n$$

соответствует число

$$(-1)^{a_0}\left(\frac{a_1}{2} + \frac{a_2}{2^2} + \ldots + \frac{a^n}{2^n}\right),$$

а в машине с плавающей запятой коду

$$\underbrace{p_0 p_1 \ldots p_p}_{\text{порядок}}\ \underbrace{a_0 a_1 \ldots a_n}_{\text{мантисса}}$$

соответствует число, порядок которого равен

$$p = (-1)^{p_0}(p_1 \cdot 2 + p_2 \cdot 2^2 + \ldots + p_r \cdot 2^r)$$

и цифровая часть равна

$$(-1)^{a_0}\left(\frac{a_1}{2} + \frac{a^2}{2^2} + \ldots + \frac{a_n}{2^n}\right).$$

ЭВЦМ основаны прежде всего на принципе адресности. Согласно этому принципу закодированные данные, необходимые для решения задач, помещаются в ячейки ЗУ, каждой из которых приписан некоторый адрес.

Инструкции-команды о выполнении операций над преобразуемой информацией составляются с помощью указаний на адреса, в которых она хранится. Для простоты в качестве адресов принимаются числа 1, 2, ..., N (N характеризует объем ЗУ).

Благодаря этому программа для решения некоторой задачи при фиксированных исходных данных становится пригодной для решения любой другой такой задачи при других исходных данных: для этого достаточно последние разместить соответственно в тех же ячейках. Таким образом, любая программа ЭВЦМ является программой для решения некоторого класса задач.

Как правило, одна и та же ячейка ЗУ может служить для хранения кодов чисел или кодов управляющих сигналов-команд.

ЗУ обычно состоит из двух частей—внутренней и внешней. Внутренняя память отличается от внешней огромной скоростью работы, т. е. скоростью, с которой могут быть извлечены или помещены в нее коды, и поэтому предназначается для автоматического использования непосредственно в АУ.

Ячейки внутренней памяти бывают двух видов — пассивные (постоянные) и активные (переменные, оперативные).

Из ячеек пассивной и активной памяти посредством УУ (в соответствии с программой) могут извлекаться коды и отсылаться в АУ для выполнения над ними операций, причем содержимое этих ячеек сохраняется. Результаты вычислений могут отсылаться только в ячейки активной памяти, стирая при этом прежде хранившиеся там коды.

Емкость внутренней части запоминающего устройства (количество отдельных кодов, которое может одновременно храниться в нем) в значительной мере определяет гибкость машины по отношению к тому кругу задач, которые возможно решать на ней. Однако объем внутренней памяти ограничен. Поэтому в современных ЭВЦМ обычно имеется большая внешняя память (иногда нескольких видов). Чаще всего — это «магнитная» память на магнитных барабанах, магнитных лентах или магнитной проволоке.

Внешняя память непосредственно с АУ не связана; коды, находящиеся в ней, согласно программе поступают группами в активную память, а также из активной памяти могут передаваться в нее группами. Время выборки кодов из внешней памяти относительно велико. Коды, зафиксированные во внешней памяти, могут храниться длительное время.

Во многих случаях в зависимости от выполняемой операции подразумевается участие в ней специальных программных регистров (модификации адресов, триггеров признаков и др.). Наличие тех или иных из них в машинах определяет специфику выполнения операций и, следовательно, специфику машинных кодов.

В связи с различием в кодах существующих машин и отсутствием единого математического языка программирования, не зависящего от их технических особенностей, существенно ограничивается возможность взаимного обмена информацией между коллективами, работающими на различных машинах. В то же время принцип адресности, благодаря которому машинные программы для решения конкретных задач превращаются в алгоритмы для решения классов задач, может быть расширен до уровня, допускающего единый для всех машин язык математического описания программ, названный адресным программированием.

Процесс подготовки задач к их постановке на машине — программирование — включает установление исходного распределения информации по адресам, а также всех промежуточных (и окончательного) ее распределений при реализации временной последовательности элементарных шагов, к которым сведено решение задачи. Эта особенность работы при решении задач на ЭВЦМ, общая для всех машин, положена в основу адресного программирования.

Исходным материалом для кодирования информации о задаче, вводимой в ЭВЦМ, служит некоторое множество двоичных чисел — кодов. Подготовка информации о задаче для ЭВЦМ состоит в кодировании отдельных элементов информации двоичными кодами и в размещении этих кодов в ячейках ЗУ машины в определенном порядке. При этом каждой ячейке ЗУ приписан адрес, который также определяется двоичным кодом.

Элементы информации (коды) в ЭВЦМ в процессе работы машины могут находиться в различных регистрах машины (например, в регистрах АУ, в сумматоре, в ре-

гистре команд УУ, в счетчике команд и т. д.) Среди этих регистров для нас существенно выделить те, которые переносят информацию от команды к команде. Такие регистры называют программными. Будем считать, что и программные регистры ЭВЦМ так же, как и ячейки ЗУ, имеют некоторые адреса [23].

Очевидно, что можно рассматривать элементы информации, содержащиеся в ЭВЦМ, адреса регистров, содержащих их, как коды одной и той же системы кодов. Поэтому размещение чисел, которыми кодируются отдельные элементы информации ЭВЦМ, можно рассматривать как задание соответствия между кодами — адресного отображения.

В соответствии с существующей интерпретацией кодов в ЭВЦМ назовем код a адресом кода $'a$, определяемого отображением $S : Sa = 'a$. При этом код $'a$ называется содержимым адреса a.

Если область определения отображения S не совпадает с множеством кодов, то для кода e, не входящего в область определения S, можно условно принять $'e = {}'|e|$, где через $|e|$ обозначена некоторая (определенная) часть кода e. Так, в машине «Киев», в которой множество всех допустимых кодов совпадает с множеством 41-разрядных двоичных чисел, а множество допустимых адресов — с отрезком натурального ряда чисел от 0 до 2047, принято в качестве $|e|$ считать код, содержащийся в разрядах, соответствующих II адресу кода e.

В одноадресной машине «Урал», в которой множество допустимых кодов совпадает с множеством 36-разрядных двоичных чисел, а множество допустимых адресов то же, что и у машины «Киев», удобно считать, что $|e|$ является кодом, содержащимся в адресной части ячейки.

Применяя последовательно адресное отображение, приходим к понятию ранга адреса.

Адресное отображение S определяет состояние ЭВЦМ в каждый момент времени.

В соответствии с этим изменение состояния ЭВЦМ зависит от преобразования отображения S. Каждое новое состояние ее является результатом преобразования кодов и передачи их из одних регистров в другие.

Таким образом, принцип адресности, принятый в современных ЭВЦМ, полностью соответствует принципу, положенному в основу адресного языка. В машинах материально осуществляется адресное отображение. В связи

с этим получают наглядный смысл символы, введенные в первой главе книги.

Так, запись

$$'a = b$$

в применении к машине имеет смысл: «в ячейке ЗУ (или регистре) с номером a помещен код b».

Запись

$$b \Rightarrow a$$

означает: поместить код b по адресу a, т. е. в ячейку ЗУ (или регистр) с номером a. Здесь a и b могут быть значениями адресных функций.

Штрих-операция приобретает смысл извлечения кодов, записанных в ячейках и регистрах машины; адресное отображение — распределение этих кодов в памяти.

4. ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ

Универсальные ЭВЦМ называют машинами с программным управлением. При решении каждой конкретной задачи управление работой машины осуществляется в соответствии с алгоритмом ее решения, представленным на языке данной машины в виде специальной записи, называемой программой.

Набор команд, составляющих программу, вводится в ЗУ машины; тем самым программа превращается в серию управляющих кодов. Для автоматической работы и правильного выполнения программы необходимо ввести в устройство управления начальную команду — первый управляющий код. После этого машина включается в автоматическую работу (специальной пусковой кнопкой); с этого момента вся работа машины управляется кодами программы вплоть до выполнения специального приказа Остановки или до остановки машины вручную (нажатием кнопки Остановка).

Последовательность управляющих кодов — программа — может насчитывать сотни или тысячи отдельных кодов (команд); по этой программе машина может выполнять сотни тысяч и даже миллионов операций, так как каждый из управляющих кодов может в процессе выполнения решения задачи использоваться многократно в различных последовательностях. Такая возможность закладывается в программу при ее составлении.

В любом случае код отдельной команды должен содержать информацию:

1) о виде (коде) операции, которую нужно выполнить по данной команде;

2) о коде или кодах, над которыми будет выполняться эта операция;

3) о месте хранения получаемого результата;

4) о коде команды, которую нужно выполнить в следующий момент.

Необходимо, чтобы по информации, содержащейся в приказе, в процессе его выполнения однозначно определялись все пп. 1—4.

Согласно принципу адресности полная информация о кодах (п. 2—4) может сообщаться указанием их адресов. Что касается вида операции, которую необходимо выполнить по данной команде (п. 1), то и эта информация также кодируется определенным образом; обычно код операции указывается непосредственно в каждой команде.

Способ задания информации в команде определяется особенностями конкретной машины. Машины разных типов имеют различные наборы выполнимых элементарных операций и отличаются способами их записи — кодирования. Эти способы записи представляют собой конкретные машинные языки.

Совокупность приказов, необходимых для решения данной задачи, занумерованная согласно их размещению во внутренней памяти конкретной машины, вместе с занумерованными исходными данными (включая постоянные, встречающиеся в решении задачи) представляет собой программу задачи.

Помимо приказов, преобразующих коды, для автоматического решения задачи на машине требуются специальные управляющие приказы, обеспечивающие обращение к внешним устройствам (ввода, вывода, внешней памяти) и реализующие предусмотренную программой последовательность выполнения операций.

Действие устройства управления в каждый данный момент времени определяется приказом, находящимся в его регистре K, и завершается засылкой соответствующего кода-приказа из ЗУ в этот регистр, т. е. автоматическим переходом к выполнению следующего приказа. В целях контроля работы в машинах помимо автомати-

ческого управления устанавливается система ручного управления для выполнения программы (или ее части) по отдельным приказам. Начальная команда при автоматическом выполнении всей программы выполняется также с помощью ручного управления, т. е. происходит запуск машины на автоматическую работу.

В зависимости от порядка выполнения команд машины делятся на машины с последовательным (естественным) и вынужденным выполнением приказов.

В машинах первого вида после выполнения k-го приказа выполняется $k+1$-й приказ или, как говорят, передается управление $k+1$-му приказу и т. д. Исключение составляют так называемые приказы передачи управления, при выполнении которых в зависимости от результатов предшествующих вычислений осуществляется переход к определенным командам программы.

Адресность машины определяется числом возможных указаний — адресов в одном приказе. В машинах с принудительным выполнением операций (приказов) в каждом из приказов выделяется дополнительный адрес, где указывается номер приказа, которому должно быть передано управление после выполнения данного приказа.

Из сказанного ясно, что одноадресные машины должны работать по принципу последовательного выполнения приказов и иметь в АУ один или несколько регистров для хранения промежуточных результатов. Арифметические действия при этом расчленяются на несколько этапов таким образом, чтобы каждая элементарная операция выполнялась только над одной величиной, взятой из ЗУ. В связи с этим в одноадресных машинах список элементарных операций должен содержать, например, такие операции, как взятие числа из ячейки α и отсылка его в регистр АУ с предварительной очисткой этого регистра ($P_1\alpha$); прибавление числа, содержащегося в ячейке α, к содержимому регистра АУ ($P_2\alpha$); взятие числа из регистра АУ и отсылка его в ячейку α ($P_3\alpha$) и т. д. Так, для выполнения операции «сложить число, хранящееся в ячейке α, с числом, хранящимся в ячейке β, и поместить результат в ячейку γ» следует поместить в программу три приказа:

$$P_1\alpha$$
$$P_2\beta$$
$$P_3\gamma.$$

В машинах с последовательным выполнением приказов и трехадресным кодом для той же цели понадобится только один приказ

$$P\alpha\beta\gamma.$$

Вместе с тем, для сложения чисел, хранящихся в ячейках $\alpha_1, \alpha_2, \ldots, \alpha_n$, с помещением результата в ячейку γ в одноадресной машине потребуется $n+1$ приказов, а в трехадресной — $n-1$ приказов. В среднем программы для трехадресных машин примерно вдвое короче соответствующих программ для машин одноадресных.

С точки зрения математика каждая конкретная машина характеризуется своим машинным языком. Однако в связи с линейностью множества машинных адресов, каждый такой язык представляет собой адресный язык (уровня условных адресов) с рядом наложенных на него ограничений и особенностей — определенный стиль языка. К таким ограничениям относятся следующие*:

1) содержимое может быть только кодом, длина которого ограничена длиной ячейки ЗУ;

2) адресами могут быть только номера (коды) ячеек ЗУ и некоторые специальные регистры;

3) ограничен набор элементарных операций машины. В связи с этим ограничен набор допустимых адресных формул;

4) множество возможных меток должно совпадать с множеством возможных адресов, так как команды помещаются обычно в тех же ячейках ЗУ, что и числа и тем самым оказываются помеченными номерами ячеек ЗУ;

5) метки в машинном алгоритме должны быть упорядочены (это замечание не касается машин с принудительным порядком выполнения команд).

При соблюдении перечисленных условий адресный алгоритм превращается в машинный, а при перекодировке его в принятую для данной машины систему команд — программу. Специально указывается начальная метка, которой соответствует начальный управляющий код, а метке ! отвечает команда останова.

---

* Заметим, что понятие адреса может быть обобщено. Учитывая удобства кодирования, адресами можно считать части ячеек или их группы.

ГЛАВА III

АДРЕСНОЕ ПРОГРАММИРОВАНИЕ

1. АДРЕСНЫЙ ЯЗЫК И ПРОГРАММИРОВАНИЕ

Задача математика, разрабатывающего программу для машины, состоит в расчленении предстоящих вычислений на последовательность отдельных элементарных операций. В отличие от человека, машина требует определения алгоритма, в формулировку которого должны быть включены дополнительные указания о том, где будут храниться те или иные величины, например исходные данные, результаты промежуточных вычислений и др. Эта особенность решения задач на машинах формулируется в виде принципа адресности, согласно которому отдельные элементы информации, преобразуемой в задаче, размещаются по адресам (в отдельных ячейках или участках ячеек). Машинная запись алгоритма, т. е. реализующая его программа, зависит от исходного и всех промежуточных отображений информации на множество адресов.

Запись алгоритма с явным указанием всех промежуточных стадий преобразования исходного отображения не только громоздка, но весьма часто просто невозможна, поскольку мы имеем дело с алгоритмами, общий ход работы которых не определен заранее и зависит от условий отдельной конкретной задачи или от ее промежуточных результатов (циклические процессы с числом циклов, определяемым в процессе вычислений, и пр.).

Запись алгоритма в виде, допускающем автоматическое выполнение его на машине, может быть получена только после выбора определенного формального (условного) языка. Именно такие способы описания алгоритмических процессов, которые не зависят от конкретных особенностей ЭВЦМ и допускают автоматическую реализацию на ЭВЦМ, следует называть программированием. Перевод

алгоритма с общего формального языка на язык конкретной машины осуществляется сравнительно легко и может быть автоматизирован путем создания соответствующих программирующих программ (трансляторов).

Одним из языков, удовлетворяющих указанным требованиям, является адресный язык. Благодаря близости его к общепринятому символическому языку математики при записи алгоритмов в нем не возникает никаких специфических, связанных именно с языком трудностей.

В то же время специальные понятия и символы адресного языка отражают общие принципы, принятые во всех современных вычислительных машинах. Поэтому алгоритм, записанный на этом языке, удобен для машинного осуществления и близок к программе любой машины.

Несмотря на то, что адресный язык основан на принципе адресности, тем не менее, к упорядоченности множества адресов не предъявляется никаких требований и адресное отображение может задаваться на произвольном множестве адресов. Для задач, по своему существу связанных с упорядоченностью элементов, могут вводиться операции следования, не описанные алгоритмически.

Уровень упорядоченности адресов, уровень алгоритмизации введенных в них операций следования можно считать уровнем языка.

Можно определить следующие три уровня адресного языка [22].

1. Общеалгоритмический уровень. Принимается наиболее естественное для данной задачи множество адресов и в тех случаях, когда этого требует задача, вводятся операции следования, описываемые общематематическими средствами (например, с помощью индексов). Так адреса компонент вектора естественно обозначить через a_i; при этом значение i-й компоненты равно $'a_i$. Запись в адресном языке этого уровня простейшего алгоритма сложения двух векторов n-го порядка будет иметь следующий вид

$$B \ \ldots \ \textbf{Ц}\{1\,(1)\,n \to i\}\,\alpha$$
$$'a_i + 'b_i \Rightarrow c_i$$
$$\alpha \ \ldots \ \textit{Я}.$$

На этом уровне адресный язык наиболее близок к АЛГОЛУ [15].

2. **Уровень условных адресов.** Предполагается упорядочение отдельных массивов адресов, обрабатываемых алгоритмом. Обычно массивы представляют собой арифметические последовательности адресов, первый (или нулевой) из которых задается. Последовательности, определяемые разными начальными адресами, предполагаются непересекающимися. Операция следования описывается таким образом уже алгоритмически. Например, операция следования по индексам для элементов матрицы, расположенных по строкам, начиная с адреса $\alpha + 1$, имеет вид $\alpha + (i-1)n + j$ (где n — порядок матрицы).

На этом уровне адреса упорядочиваются, только исходя из требований задачи. Взаимная же упорядоченность массивов и другие вопросы, связанные с фактическим распределением памяти, не решаются. Тем не менее уже на этом уровне решаются практически все основные вопросы, относящиеся к области программирования для конкретных машин, например вопрос о выборе параметров, о рациональном распределении памяти, о рациональном построении циклов, экономии рабочих ячеек и др.

Этот уровень языка используется и в данной книге для описания адресных программ.

Кроме того, как правило, принимается стандартный способ расположения массивов в так называемые α-последовательности. Условимся знаки арифметических операций, с помощью которых описываются операции следования в множествах адресов, т. е. знаки арифметических действий, результатом выполнения которых является адрес, записывать в виде $\oplus$, $\ominus$, $\otimes$, $\odot$.

Таким образом, ранее приведенный алгоритм на этом уровне может быть записан в виде

$$B \ldots \text{Ц}\{1\,(1)\,n \to I\}\,\alpha$$
$$'(\alpha \oplus I) \div '(\beta \oplus I) \Rightarrow \gamma \oplus I$$
$$\alpha \ldots \text{Я}.$$

3. Уровень конкретных адресов. Предполагается, что множество адресов, исключая программные регистры, полностью упорядочено. Обычно имеется в виду выполнение данного алгоритма на конкретной машине, в связи с чем и решаются вопросы определения истинных операций следования.

Если разместить одну компоненту в одной машинной ячейке, то приведенный выше пример выглядит, например, так:

$$B \ldots \textbf{Ц}\{1\,(1)\,7 \to I\}\,\alpha$$
$$'(100 \oplus I) + '(200 \oplus I) \Rightarrow 300 \oplus I$$
$$\alpha \ldots \textbf{Я}.$$

Не следует забывать, что один абстрактный адрес может занимать несколько ячеек памяти. Например, на машине «Урал» компоненты можно располагать в длинных ячейках, т. е. в парах коротких. Тогда тот же алгоритм запишется так:

$$B \ldots \textbf{Ц}\{0\,(2)\,12 \to I\}\,\alpha$$
$$'(100 \oplus I) + '(200 \oplus I) \Rightarrow 300 \oplus I$$
$$\alpha \ldots \textbf{Я}.$$

Для выполнения операций на «Урале» с удвоенной точностью расположим числа в четырех коротких ячейках, т. е. получим другую операцию следования. При реализации этих алгоритмов на машинах принимаются меры для выполнения операции над содержимым адреса; в последнем алгоритме это достигается употреблением признака полноты ячейки. Этот вопрос выходит за пределы адресного языка и поэтому здесь не рассматривается.

Абстрактный адрес может составлять и часть ячейки памяти.

В адресном языке заложена возможность перехода от уровня к уровню, начиная от самого абстрактного алгоритмического языка и кончая полным распределением адресов для данной машины. Это делает адресный язык полностью пригодным в качестве входного языка для программирующих программ (ПП). Три описанных выше уровня, конечно, не исчерпывают всех возможных промежуточных. В зависимости от уровня принятого входного языка создаются более и менее сложные ПП. Такие ПП для машин «Урал-1», «Киев» и др. уже работают в Институте кибернетики АН УССР.

Свободно используя средства адресного языка (речь идет о применении его операторов), можно записать один и тот же алгоритм самыми различными способами. Перевод со свободного адресного языка на язык конкретной машины довольно труден. Поэтому вводятся так называе-

мые стили языка. Стиль предполагает такие ограничения в применении выразительных средств свободного языка, которые делают перевод с данного стиля на язык конкретной машины более простым.

Стиль языка не связан с его уровнем. Однако ограничения, налагаемые стилем на язык, не должны лишать его универсальности.

Перевод с общего языка в определенный стиль его возможен на любом уровне, но правила такого перевода должны быть строго сформулированы. Ограничения, налагаемые на язык, могут быть настолько сильными, что станет возможным поформульный перевод записи алгоритма в команды машины, т. е. запись алгоритма в данном стиле будет отличаться от машинной программы только кодировкой. В этом смысле язык любой машины является адресным.

Таким образом, благодаря широте и гибкости адресного языка можно произвольно выбирать границу, до которой необходимо вручную доводить программы задач с тем, чтобы дальнейшую их обработку проводила программирующая программа. Более детально этот вопрос рассматривается в гл. V.

Любой алгоритм или конечное число алгоритмов содержит конечное число операций. Следовательно, можно построить машину ИАВМ (идеальную адресную вычислительную машину), для которой данный алгоритм и все алгоритмы, использующие те же действия, являются программой. Для этого [23] нужно включить в набор элементарных операций машины все встречающиеся в алгоритме действия и обеспечить кодировку команд, принятую в адресном языке.

Реальные универсальные вычислительные машины отличаются от ИАВМ прежде всего тем, что набор их элементарных операций, хотя и является полным, ограничен небольшим числом.

Превращение адресной программы в программу конкретной машины является моделированием на этой машине работы ИАВМ.

При переходе от адресной программы к программе конкретной машины необходимо учитывать, что каждая конкретная вычислительная машина характеризуется объемом и структурой памяти, а также принятой кодировкой информации.

Моделирование на ИАВМ работы любой реальной универсальной вычислительной машины сводится просто к выбору из элементарных операций ИАВМ только тех, которые входят в набор данной реальной машины. Практически это означает ограничение числа допустимых операций (формул адресного языка). Для превращения в программу данной реальной машины написанная при соблюдении такого ограничения адресная программа нуждается только в перекодировке.

Таким образом, адресное программирование на уровне условных адресов позволяет без учета особенностей конкретной машины написать всякий алгоритм в виде, удобном для любой машины. Одновременно при составлении адресной программы решаются все вопросы, связанные с рассматриваемой задачей и общими для всех машин особенностями ее решения. Поэтому на уровне адресного программирования возможен обмен информацией для различных машин.

2. ЗАДАЧА ОБОЗРЕВАНИЯ ИНФОРМАЦИИ

Говорят, что элемент информации обозревается алгоритмом, если его адрес некоторого ранга содержится в адресной программе этого алгоритма.

Адресное программирование, главным образом, сводится к построению схем обозревания информации.

Схемой обозревания последовательности кодов

$$a_0,\ a_1,\ a_2,\ \ldots,\ a_n \tag{3.1}$$

называют циклическую адресную программу, на i-м из циклов которой обозревается i-й элемент последовательности (3.1).

При решении любой задачи требуется обозревание относящейся к ней информации. Это легко достигается в алгоритмах, если информация невелика и связь между элементами множества, определяющая операции следования в ней, проста. При большом объеме информации и сложных операциях следования для составления схем обозревания необходимо использовать адреса второго и высших рангов.

Рассмотрим схемы обозревания информации, которые чаще всего используются в программах широкого круга математических и логических задач, и одновременно произведем некоторую классификацию этих схем.

3. СХЕМЫ ОБОЗРЕВАНИЯ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Пусть в последовательности адресов

$$\alpha_0,\ \alpha_1,\ \ldots,\ \alpha_n\ (\text{'}\alpha_i = a_i,\ i = 0,\ 1,\ \ldots,\ n) \qquad (3.2)$$

определена операция следования C:

$$\alpha_{(i+1)} = C\alpha_i\ (i = 0,\ 1,\ \ldots,\ n-1).$$

Для построения схемы обозревания последовательности кодов (3.1) введем адрес φ для начального элемента последовательности (3.2):

$$\text{'}\varphi = \alpha_0,$$

т. е. φ — есть адрес второго ранга. или фиксатор кода a_0:

$$^2\varphi = a_0.$$

Тогда циклическая программа для обозревания последовательности кодов $a_1,\ a_2,\ \ldots,\ a_n$ может быть записана в виде

$$\begin{array}{c} B\ \ldots\ C^1\varphi \Rightarrow \varphi \\ \Phi\,(^2\varphi) \\ \boldsymbol{P}\,\{\text{'}\varphi \neq \alpha_n\}\ B \downarrow \mathcal{A}. \end{array} \qquad (3.3)$$

Здесь через $\Phi(^2\varphi)$ обозначена некоторая адресная программа, включающая адрес второго ранга φ. Входная (начальная) формула отмечена меткой B.

Проследим за работой программы. Вначале $\text{'}\varphi = \alpha_0$. В результате выполнения первой строки получим

$$\text{'}\varphi = \alpha_1,$$

так как

$$C\text{'}\varphi = C\alpha_0 = \alpha_1.$$

При первом выполнении второй строки будет реализовываться программа

$$\Phi\,(^2\varphi) = \Phi\,(\text{'}\alpha_1) = \Phi\,(a_1).$$

Далее проверяется условие $\text{'}\varphi \neq \alpha_n$, при этом $\text{'}\varphi = \alpha_1$ и, если $n > 1$, т. е. проверяемое условие выполняется, совершается переход к первой формуле, помеченной меткой B. В результате выполнения этой формулы получим $\text{'}\varphi = \alpha_2$.

После n-кратного повторения программы $\text{'}\varphi$ станет равно α_n и условие $\text{'}\varphi \neq \alpha_n$ окажется невыполнимым.

В результате по метке $Я$ предикатной формулы произойдет остановка.

Применение операции следования C, заданной для множества адресов последовательности, к содержимому фиксатора этой последовательности называют сдвигом фиксатора.

В проведенной схеме (3.3) элемент информации a_0 программой $\Phi(^2\varphi)$ не обозревается. Если в этой программе переставить местами первую и вторую строки, то получим схему:

$$\begin{array}{c} B \ \ldots \ \Phi(^2\varphi) \\ C'\varphi \Rightarrow \varphi \\ \boldsymbol{P}\{'\varphi \neq a_n\}\, B \downarrow Я. \end{array} \qquad (3.4)$$

Программой (3.4) обозреваются коды

$$a_n,\ a_1,\ \ldots,\ a_{n-1}.$$

Если, кроме того, принять $'\varphi 1 = a_n$, то программы (3.3) и (3.4) принимают стандартный вид, не зависящий от исходного адресного отображения:

$$\begin{array}{c} B \ \ldots \ C'\varphi \Rightarrow \varphi \\ \Phi(^2\varphi) \\ \boldsymbol{P}\{'\varphi \neq {}'\varphi 1\}\, B \downarrow Я \end{array} \qquad (3.5)$$

или

$$\begin{array}{c} B \ \ldots \ \Phi(^2\varphi) \\ C'\varphi \Rightarrow \varphi \\ \boldsymbol{P}\{'\varphi \neq {}'\varphi 1\}\, B \downarrow Я. \end{array} \qquad (3.6)$$

Программы (3.3), (3.4) или (3.5), (3.6) связаны с исходной информацией (3.1) тем, что они включают операцию следования C, заданную для последовательности адресов (3.2). Кроме того, в программе используются два адреса второго ранга φ и $\varphi 1$; φ обозревает всю последовательность (3.1); а $\varphi 1$ — ее последний элемент (3.1).

Посмотрим, что произойдет с программами (3.3) и (3.4), если в них принять вторую формулу в качестве начальной.

Программы (3.3) и (3.4) перепишутся в виде

$$\begin{array}{c} K \ \ldots \ C'\varphi \Rightarrow \varphi \\ B \ \ldots \ \Phi(^2\varphi) \\ \boldsymbol{P}\{'\varphi \neq {}'\varphi 1\}\, K \downarrow Я, \end{array} \qquad (3.7)$$

и

$$K \ldots \Phi(^2\varphi)$$
$$B \ldots C'\varphi \Rightarrow \varphi$$
$$\boldsymbol{P}\{'\varphi \neq '\varphi 1\} K \downarrow \mathcal{Я}. \qquad (3.8)$$

В обоих случаях понадобилась еще одна метка K, поскольку меткой B обозначена теперь вторая формула, а к первой из формул имеется переход по верхней метке предикатной формулы.

Легко проверить, что в программе (3.7) операция Φ осуществляется для всех элементов последовательности (1.1) от a_0 до a_n, а в программе (3.8) крайние элементы исключаются.

Пусть, например, последовательность адресов (3.2) упорядочена условием $\alpha_{i+1} = \alpha_i \oplus 1$, т. е. операция следования состоит в прибавлении единицы, тогда схема обозревания последовательности (1.1) будет иметь вид:

$$K \ldots '\varphi \oplus 1 \Rightarrow \varphi$$
$$B \ldots \Phi(^2\varphi)$$
$$\boldsymbol{P}\{'\varphi \neq '\varphi 1\} K \downarrow \mathcal{Я}. \qquad (3.9)$$

В этом случае $'\varphi 1 = \alpha_0 \oplus n$, в связи с чем вместо предикатной формулы, использующей отношение неравенства $'\varphi$ и $'\varphi 1$, может быть взята следующая:

$$\boldsymbol{P}\{'\varphi < '\varphi 1\} K \downarrow \mathcal{Я}.$$

Для включения в программу обозревания $\Phi(^2\varphi)$ последнего элемента последовательности (1.1) можно использовать неравенство $'\varphi \leqslant \varphi 1$ и схему обозревания (3.4):

$$B \ldots \Phi(^2\varphi)$$
$$'\varphi \oplus 1 \Rightarrow \varphi$$
$$\boldsymbol{P}\{'\varphi \leqslant '\varphi 1\} B \downarrow \mathcal{Я}. \qquad (3.10)$$

Во всех приведенных схемах требуется, чтобы к началу их работы, т. е. к моменту выхода на метку B, по адресу φ содержался адрес первого элемента обозреваемой последовательности.

Если какая-либо из приведенных схем используется в некоторой программе в качестве подсхемы и возможно ее повторное применение, то необходимо учитывать, что фиксатор последовательности φ в процессе работы программы сдвигается от первого к последнему ее элементу.

При этом возможны следующие случаи.

1. Схемой при повторном ее применении будет снова обозреваться исходный массив: в этом случае фиксатор φ

следует сделать рабочим, т. е. таким, который самой схемой устанавливается вначале в положение первого элемента и затем в процессе обозревания сдвигается к последнему элементу последовательности.

Подобные схемы называют схемами обозреваний с восстановлением.

Так, например, схема (3.10) с восстановлением может быть записана в виде:

$$\begin{array}{c} B \ \ldots \ \alpha_0 \Rightarrow \varphi \\ K \ \ldots \ \Phi(^2\varphi) \\ {}'\varphi \bigoplus 1 \Rightarrow \varphi \\ \boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\}\, K \downarrow \mathcal{A}. \end{array} \tag{3.11}$$

Схема (3.11) зависит от исходного отображения, так как она явно содержит адрес начального элемента обозреваемой последовательности. Для устранения этой зависимости введем фиксатор начального элемента ${}'\varphi_0 = \alpha_0$ и назовем его начальным фиксатором.

Тогда схема (3.11) перепишется в виде

$$\begin{array}{c} B \ \ldots \ {}'\varphi_0 \Rightarrow \varphi \\ K \ \ldots \ \Phi(^2\varphi) \\ {}'\varphi \bigoplus 1 \Rightarrow \varphi \\ \boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\}\, K \downarrow \mathcal{A}. \end{array} \tag{3.12}$$

Работа ее определяется содержимым адресов φ_0 и $\varphi 1$.

2. При повторном применении схемы обозревается группа адресов, составляющих продолжение последовательности. В этом случае для последующего применения схемы содержимое φ не должно восстанавливаться, но в схему необходимо включить операцию следования, с помощью которой фиксатор $\varphi 1$ устанавливается в положение нового конечного элемента последовательности. Так, например, если при каждом новом применении схемы обозреваются следующие n элементов последовательности и ${}'\delta = n$, то такая схема обозревания может представиться в виде (вначале ${}'\varphi = \alpha$; ${}'\varphi 1 = \alpha \bigoplus n$)

$$\begin{array}{c} B \ \ldots \ \Phi(^2\varphi) \\ {}'\varphi \bigoplus 1 \Rightarrow \varphi \\ \boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\}\, B \\ {}'\varphi 1 \bigoplus {}'\delta \Rightarrow \varphi 1 \\ \cdot \\ \cdot \\ \cdot \end{array}$$

Схемы такого типа называют схемами без восстановления.

3. При повторном применении схемы обозревается новая последовательность элементов.

В этом случае в схеме, которую можно рассматривать как подсхему некоторой основной схемы, к моменту ее использования должны быть определены содержимые адресов φ и $\varphi 1$ — фиксаторов начала и конца обозреваемых последовательностей. Например, к началу работы схемы (3.12) по адресу φ_0 должен быть заслан начальный адрес β новой последовательности, а по адресу $\varphi 1$ — конечный адрес $\beta \oplus k$.

Схемы этого вида называют схемами типа подпрограмм.

Во всех приведенных схемах (3.9—3.12) предполагалось, что $'\varphi 1 \geqslant \alpha_0$, т. е. последовательность содержит хотя бы один элемент. Соответственно этому во всех схемах программа Φ осуществлялась по крайней мере один раз для начального элемента.

Иногда встречаются схемы, в которых обозреваются последовательности с различным числом членов, в частности равным нулю.

В этом случае каждая из рассмотренных схем должна быть перестроена так, чтобы до применения программы $\Phi(^2\varphi)$ выполнялась проверка условия принадлежности рассматриваемому множеству элемента, задаваемого $^2\varphi$.

Пусть, например, условие принадлежности последовательности элемента $^2\varphi$:

$$'\varphi \leqslant '\varphi 1.$$

Тогда схема обозревания для операции следования, состоящей в прибавлении единицы, может быть записана в виде

$$\begin{array}{c} B \ \ldots \ \boldsymbol{P}\{'\varphi \leqslant '\varphi 1\} \downarrow \mathcal{A} \\ \Phi(^2\varphi) \\ '\varphi \oplus 1 \Rightarrow \varphi \\ B. \end{array} \qquad (3.13)$$

Здесь последняя строка является меткой безусловного перехода. Если для первого значения $'\varphi$ отношение $'\varphi \leqslant '\varphi 1$ не выполняется, то схема заканчивает свою работу по метке $\mathcal{A}$.

Схема (3.13) с восстановлением по начальному фиксатору запишется в виде:

$$\begin{array}{c} B \ \ldots \ {}'\varphi_0 \Rightarrow \varphi \\ K \ \ldots \ \boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\} \downarrow \mathit{Я} \\ \Phi({}^2\varphi) \\ {}'\varphi \bigoplus 1 \Rightarrow \varphi \\ K. \end{array} \tag{3.14}$$

Эта же схема (3.13) без восстановления со сдвигом на $'\delta$ элементов имеет вид

$$\begin{array}{c} B \ \ldots \ \boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\} \downarrow K \\ \Phi({}^2\varphi) \\ {}'\varphi \bigoplus 1 \Rightarrow \varphi \\ B \\ K \ \ldots \ {}'\varphi 1 \bigoplus {}'\delta \Rightarrow \varphi 1 \\ \mathit{Я}. \end{array} \tag{3.15}$$

С помощью формулы циклирования $\mathit{Ц}$ две последние схемы можно представить в более компактном виде. А именно, схеме с восстановлением (3.14) будет соответствовать схема

$$\begin{array}{c} \boldsymbol{Ц}\{{}'\varphi_0(1){}'\varphi 1 \Rightarrow \varphi\}\alpha \\ \Phi({}^2\varphi) \\ \alpha \ \ldots \ \mathit{Я}, \end{array} \tag{3.16}$$

а схеме без восстановления (3.15) — следующая:

$$\begin{array}{c} B \ \ldots \ \boldsymbol{Ц}\{{}'\varphi(1){}'\varphi 1 \Rightarrow \varphi\} K \\ \Phi({}^2\varphi) \\ K \ \ldots \ {}'\varphi 1 \bigoplus {}'\delta \Rightarrow \varphi 1 \\ \mathit{Я}. \end{array} \tag{3.17}$$

Схемы (3.15) и (3.17) отличаются только формой записи.

Сравним схемы (3.12) и (3.16). Для ${}'\varphi_0 \leqslant {}'\varphi 1$ эти схемы эквивалентны. Однако, если ${}'\varphi_0 > {}'\varphi 1$, по схеме (3.12) будет реализована программа $\Phi({}^2\varphi)$ для ${}^2\varphi = {}^2\varphi_0$; по схеме (3.16) вначале будет проверено выполнение условия ${}'\varphi_0 > {}'\varphi 1$ (шаг изменения параметра φ положителен), и при невыполнении его схема закончит работу по метке $\mathit{Я}$.

Пусть последовательность адресов образует арифметическую прогрессию с разностью d; тогда схема (3.13) перепишется в виде

$$\begin{array}{c} B \ \ldots \ \boldsymbol{P}\{'\varphi \leqslant {}'\varphi 1\} \downarrow \mathcal{A} \\ \Phi\,({}^2\varphi) \\ {}'\varphi \bigoplus \delta \Rightarrow \varphi \\ B, \end{array}$$

или, полагая ${}'\delta = d$, получаем схему

$$\begin{array}{c} B \ \ldots \ \boldsymbol{P}\{'\varphi \leqslant {}'\varphi 1\} \downarrow \mathcal{A} \\ \Phi\,({}^2\varphi) \\ {}'\varphi \bigoplus {}'\delta \Rightarrow \varphi \\ B. \end{array} \qquad (3.18)$$

Схема (3.18) является более общей по сравнению со схемой (3.13). Во-первых, она не зависит явно от разности d. Во-вторых, можно считать, что содержимое адреса δ формируется от цикла к циклу в процессе выполнения программы $\Phi({}^2\varphi)$. Таким образом, схему (3.18) можно рассматривать как схему обозревания последовательности с операцией следования, определяемой в процессе работы программы.

Программу (3.13) назовем схемой обозревания со сдвигом по адресу нулевого ранга, поскольку в ней операция следования задается сложением с адресом нулевого ранга (константой). Соответственно программу (3.18) назовем схемой обозревания со сдвигом по адресу, а ячейку δ, содержащую величину сдвига, — адресом сдвига.

Схема обозревания со сдвигом по адресу и с восстановлением по начальному фиксатору φ_0 имеет вид

$$\begin{array}{c} B \ \ldots \ {}'\varphi_0 \Rightarrow \varphi \\ K \ \ldots \ \boldsymbol{P}\{'\varphi \leqslant {}'\varphi 1\} \downarrow \mathcal{A} \\ \Phi\,({}^2\varphi) \\ {}'\varphi \bigoplus {}'\delta \Rightarrow \varphi \\ K. \end{array} \qquad (3.19)$$

С помощью формулы циклирования схема (3.19) перепишется в виде

$$\begin{array}{c} \boldsymbol{Ц}\{'\varphi_0\,('\delta)'\varphi 1 \Rightarrow \varphi\}\,\alpha \\ \Phi\,({}^2\varphi) \\ \alpha \ \ldots \ \mathcal{A}. \end{array} \qquad (3.20)$$

Содержимое адреса сдвига δ определяется программой Φ. Однако к началу работы программы $'\delta$ должно быть определено. Если $'\delta > 0$, то по фомуле циклирования проверяется условие $'\varphi_0 \leqslant '\varphi 1$, и в случае его невыполнения совершается выход по метке $\mathcal{Я}$. Аналогично при $'\delta < 0$, проверяется условие $'\varphi_0 \geqslant '\varphi 1$.

Рассмотрим вариант реализации подобных схем с помощью так называемого счетчика.

Пусть

$$\begin{gathered}\alpha_{i+1} = \alpha_i \oplus '\delta;\\ '\varphi_0 = \alpha_0;\\ '\beta 1 = \alpha_n \ominus \alpha_0.\end{gathered}$$

Значения $'\delta$ таковы, что каждый из адресов обозревается однократно. Тогда схема (3.18) может быть переписана в виде

$$\begin{gathered}B \ \ldots \ 0 \Rightarrow \beta\\ K \ \ldots \ P\{'\beta \leqslant '\beta 1\} \downarrow \mathcal{Я}\\ \Phi('('\varphi_0 \oplus '\beta))\\ '\beta \oplus '\delta \Rightarrow \beta\\ K\end{gathered} \tag{3.21}$$

или с формулой $Ц$ в виде

$$\begin{gathered}Ц\{0('\delta)'\beta 1 \Rightarrow \beta\}\alpha\\ \Phi('('\varphi_0 \oplus '\beta))\\ \alpha \ \ldots \ \mathcal{Я}.\end{gathered} \tag{3.22}$$

Если $'\delta = 1$, а $'\beta 1 = \alpha_n \ominus \alpha_0 = n$, то схема (3.22) обозревает все элементы последовательности (3.1). В этой программе φ_0 содержит неизменно адрес первого кода последовательности (3.1), т. е. является фиксатором начала последовательности (3.1).

По адресу β, который осуществляет последовательное обозревание элементов (3.1), можно определять порядковый номер (индекс) элемента последовательности (3.1).

Если $'\delta = 1$, то содержимое β равно номеру цикла работы схемы, т. е. β играет роль счетчика

$$\begin{gathered}B \ \ldots \ 0 \Rightarrow \beta\\ K \ \ldots \ \boldsymbol{P}\{'\beta \leqslant n\} \downarrow \mathcal{Я}\\ \Phi('('\varphi \oplus '\beta))\\ '\beta \oplus 1 \Rightarrow \beta\\ K,\end{gathered} \tag{3.23}$$

или

$$\begin{array}{c} \boldsymbol{Ц}\,\{0\,(1)\,n \Rightarrow \beta\}\,\alpha \\ \Phi\,('\,('\varphi \oplus '\beta)) \\ \alpha\ \ldots\ \mathit{Я}. \end{array} \qquad (3.24)$$

Приведем вариант схемы (3.23) со счетчиком β, работающим по принципу вычитания.

Положив $'\varphi 1 = \alpha_0 \oplus n$ (фиксируем последний элемент последовательности (3.1)), получим схему обозревания

$$\begin{array}{c} B\ \ldots\ n \Rightarrow \beta \\ K\ \ldots\ \boldsymbol{Р}\,\{'\beta \geqslant 0\} \downarrow \mathit{Я} \\ \Phi\,('\,('\varphi 1 \ominus '\beta)) \\ '\beta \ominus 1 \Rightarrow \beta \\ K, \end{array} \qquad (3.25)$$

или

$$\begin{array}{c} \boldsymbol{Ц}\,\{n\,(\ominus 1)\,0 \Rightarrow \beta\}\,\alpha \\ \Phi\,('\,('\varphi 1 \ominus '\beta)) \\ \alpha\ \ldots\ \mathit{Я}. \end{array}$$

Рассмотрим применение этих схем.

Пример. 1. *Вычисление и печать таблицы значений функции от ряда значений аргумента.*

Пусть ряд значений аргумента $x_1, x_2, \ldots, x_n$, для которых необходимо определить значение функции $\Phi(x)$, расположен по адресам

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots\ \alpha \oplus n$$

и, кроме того,

$$'\varphi = \alpha \oplus 1,\ '\bar{\varphi} = \alpha \oplus n.$$

Предположим, что при засылке какого-либо кода по адресу *Пч* он автоматически выдается на печать.

Адресная программа, которая решает поставленную задачу и включает схему обозревания (3.10), имеет вид

$$\begin{array}{c} B\ \ldots\ \Phi\,(^2\varphi) \rightarrow \mathit{Пч} \\ '\varphi \oplus 1 \rightarrow \varphi \\ \boldsymbol{Р}\,\{'\varphi \leqslant '\bar{\varphi}\}\ B \downarrow \mathit{Я}. \end{array}$$

Можно легко записать аналогичную схему обозревания с восстановлением и с применением формулы ***Ц***. В качестве упражнения предлагается проделать то же самое и для всех других примеров этого параграфа.

При том же адресном отображении программа со схемой (3.25) перепишется следующим образом

$$B \ldots n \ominus 1 \to \beta$$
$$K \ldots P\{'\beta \geqslant 0\} \downarrow \mathcal{A}$$
$$\Phi('('\varphi \ominus '\beta)) \to \Pi u$$
$$'\beta \ominus 1 \to \beta$$
$$K.$$

Здесь принято $'\varphi = \alpha \oplus n$.

Пример 2. *Умножение вектора с компонентами*

$$a_1,\ a_2,\ \ldots,\ a_n \tag{3.26}$$

на скаляр k.

Компоненты вектора (3.26) и скаляр k размещены соответственно по адресам

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha \oplus n;\ \gamma.$$

Компоненты результирующего вектора получаем по тем же адресам.

Введем для первой компоненты вектора (3.26) адрес второго ранга $'_4 = \alpha \oplus 1$ и пусть $'\bar{\varphi} = \alpha \oplus n$; тогда получим программу

$$B \ldots {}^2\varphi \times '\gamma \to '\varphi$$
$$'\varphi \oplus 1 \to \varphi$$
$$P\{'\varphi \leqslant '\bar{\varphi}\}\ B \downarrow \mathcal{A}.$$

Если положить $'\beta = 1$; $'\beta 1 = n$, то получим второй вариант программы со счетчиком по принципу сложения:

$$B \ldots {}' ('\varphi \oplus '\beta) \times '\gamma \to '\varphi \oplus '\beta$$
$$'\beta \oplus 1 \to \beta$$
$$P\{'\beta \leqslant '\beta 1\}\ B \downarrow \mathcal{A},$$

если же положить $'\varphi = \alpha \oplus n$; $'\beta = n \ominus 1$, получаем программу со счетчиком по принципу вычитания:

$$B \ldots {}' ('\varphi \ominus '\beta) \times '\gamma \to '\varphi \ominus '\beta$$
$$'\beta \ominus 1 \to \beta$$
$$P\{'\beta \geqslant 0\}\ B \downarrow \mathcal{A}.$$

Пример 3. *Вычисление таблицы значений функций* $f(x)$ *для последовательно вычисляемых значений аргумента* x, *т. е.*

$$x_{i+1} = x_i + h \quad (i = 0,\ 1,\ \ldots,\ n-1)$$

с размещением полученных значений по адресам $\alpha \oplus 1, \ldots, \alpha \oplus n$ *(с запоминанием).*

Пусть $'\varphi = \alpha \oplus 1$; $'\gamma_1 = x_0$; $'\gamma_2 = h$; $'\varphi 1 = \alpha \oplus n$, тогда получим программу

$$B \ldots P\{'\varphi \leqslant '\varphi 1\} \downarrow \mathcal{A}$$
$$'\gamma_1 + '\gamma_2 \to \gamma_1$$
$$f('\gamma_1) \to '\varphi$$
$$'\varphi \oplus 1 \to \varphi;\ B.$$

Полагая $'\varphi = \alpha \oplus n$ и $'\beta = n \ominus 1$, получим

$$B \ldots \boldsymbol{P}\{'\beta \geqslant 0\} \downarrow \mathcal{A}$$
$$'\gamma_1 + '\gamma_2 \to \gamma_1$$
$$f('\gamma_1) \to '\varphi \ominus '\beta$$
$$'\beta \ominus 1 \to \beta$$
$$B.$$

В приведенных программах (при $h > 0$) в качестве предикатной может быть использована следующая формула:

$$\boldsymbol{P}\{'\gamma_1 \leqslant \bar{'a}\} \downarrow \mathcal{A},$$

если принять

$$\bar{'a} = x_n = x_0 + nh.$$

Запишем эти программы с помощью формулы $\boldsymbol{Ц}$:

$$\boldsymbol{Ц}\{'\varphi\,(1)'\,\varphi 1 \to \varphi\}\ \alpha$$
$$'\gamma_1 + '\gamma_2 \to \gamma_1,\ f(\gamma_1) \to '\varphi$$
$$\alpha \ldots \mathcal{A}$$

или, если $'\varphi = \alpha \oplus n$, $'\beta = n \ominus 1$, получим

$$\boldsymbol{Ц}\{'\beta\,(\ominus 1)\,0 \to '\beta\}\ \alpha$$
$$'\gamma_1 + '\gamma_2 \to \gamma_1,\ f('\gamma_1) \to '\varphi \ominus '\beta$$
$$\alpha \ldots \mathcal{A}.$$

Пример 4. *Вычисление суммы*

$$s = \sum_{i=1}^{n} a_i \tag{3.27}$$

или произведения

$$\pi = \prod_{i=1}^{n} a_i.$$

Пусть величины a_i размещаются по адресам $\alpha \oplus 1, \ldots, \alpha \oplus n$, т. е. $'(\alpha \oplus i) = a_i$, а результат получается по адресу β.

Положим

$$'\varphi = \alpha \oplus 1;\ '\varphi 1 = \alpha \oplus n.$$

Получаем программу для вычисления суммы

$$0 \to \beta$$
$$K \ldots {}'\beta + {}^2\varphi \to \beta$$
$$'\varphi \oplus 1 \to \varphi$$
$$\boldsymbol{P}\{'\varphi \leqslant '\varphi 1\}\ K \downarrow \mathcal{A}$$

и программу для вычисления произведения

$$1 \to \beta$$
$$K \ldots {}'\beta \times {}^2\varphi \to \beta$$
$$'\varphi \oplus 1 \to \varphi$$
$$\boldsymbol{P}\{'\varphi \leqslant '\varphi 1\}\ K \downarrow \mathcal{A}.$$

В дальнейшем, где из текста будет ясно, о каком исходном и результативном адресном отображении идет речь, их описание в программах будем опускать.

Поскольку входными формулами в последних двух программах являются первые формулы, к которым в дальнейшем отсутствует переход, метка H опущена.

Пусть теперь члены суммы (3.27) будут значениями некоторой функции f от ряда значений аргумента: $a_1, a_2, \ldots, a_n$. Тогда программа перепишется в виде

$$
\begin{gathered}
0 \to \beta \\
K \ldots {}'\beta + f({}^2\varphi) \to \beta \\
{}'\varphi \oplus 1 \to \varphi \\
\boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\}\, K \downarrow \mathcal{Я}.
\end{gathered}
$$

Ввиду для аргумента функции f стандартный адрес a, предыдущую программу можно переписать так:

$$
\begin{gathered}
0 \to \beta \\
K \ldots {}^2\varphi \to a \\
{}'\beta + f({}'a) \to \beta \\
{}'\varphi \oplus 1 \to \varphi \\
\boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\}\, K \downarrow \mathcal{Я}.
\end{gathered}
$$

В машинах операция извлечения содержимого по адресу второго ранга ${}^2\varphi$ более сложна, чем операция извлечения содержимого по адресу первого ранга. Поэтому хотя в данном алгоритме на одну строку больше, чем в предыдущем, при сложной зависимости f от своего аргумента последняя программа может оказаться более выгодной, чем предыдущая.

Пример 5. *Определение суммы первых из n членов последовательности $a_1, a_2, \ldots, a_n$, которые принадлежат множеству M (например, суммы первых положительных членов).*

Пусть, как и в примере 4, ${}'(\alpha \oplus i) = a_i$, искомый результат помещается по адресу β, ${}'\varphi = \alpha \oplus 1$ и ${}'\varphi 1 = \alpha + n$. Тогда получаем программу

$$
\begin{gathered}
0 \to \beta \\
K \ldots \boldsymbol{P}\{{}^2\varphi \overline{\in} M\}\, \mathcal{Я} \\
\boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\} \downarrow \mathcal{Я} \\
{}'\beta + {}^2\varphi \to \beta \\
{}'\varphi \oplus 1 \to \varphi \\
K \ldots
\end{gathered}
$$

С формулой циклирования эта программа принимает вид

$$
\begin{gathered}
0 \to \beta \\
\boldsymbol{Ц}\{{}'\varphi(1)\, \boldsymbol{P}\{{}^2\varphi \in M\} \to \varphi\}\, K \\
\boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\} \downarrow \mathcal{Я} \\
{}'\beta + {}^2\varphi \to \beta \\
K \ldots \mathcal{Я},
\end{gathered}
$$

или

$$
\begin{array}{c}
0 \to \beta \\
\boldsymbol{Ц}\,\{'\varphi\,(1)\,\boldsymbol{P}\,\{(^2\varphi \in M) \wedge ('\varphi \leqslant '\varphi 1)\} \to \varphi\}\,K \\
'\beta + {}^2\varphi \to \beta \\
K\ \ldots\ \mathcal{Я},
\end{array}
$$

или

$$
\begin{array}{c}
0 \to \beta \\
\boldsymbol{Ц}\,\{'\varphi\,(1)\,\boldsymbol{P}\,\{'\varphi \leqslant '\varphi 1\} \to \varphi\}\,K \\
\boldsymbol{P}\,\{^2\varphi \in M\} \downarrow \mathcal{Я} \\
'\beta + {}^2\varphi \to \beta \\
K\ \ldots\ \mathcal{Я}.
\end{array}
$$

Пример 6. *Вычисление значения многочлена*

$$f(x) = a_0x^n + a_1x^{n-1} + \ \ldots\ + a_n.$$

Рассмотрим программу вычисления значения многочлена по следующей схеме:

$$f(x) = (\ldots((a_0x + a_1)\,x + \ \ldots\ + a_{n-1})\,x + a_n.$$

Пусть заданные величины $a_0, a_1, \ldots, a_n$; x расположены соответственно по адресам

$$\alpha,\ \alpha \oplus 1,\quad \alpha \oplus 2,\ \ldots,\ \alpha \oplus n;\quad \beta,$$

и

$$'\varphi = \alpha;\quad '\delta 1 = n;\quad '\delta = 0.$$

Вычисляемое значение функции поместим по адресу γ.

Тогда программа (без восстановления) имеет вид

$$
\begin{array}{c}
0 \to \gamma \\
K\ \ldots\ '\gamma \times '\beta + '\,('\varphi \oplus '\delta) \to \gamma \\
'\delta \oplus 1 \to \delta \\
\boldsymbol{P}\,\{'\delta \leqslant '\delta 1\}\,K \downarrow \mathcal{Я}.
\end{array}
$$

Пример 7. *Нахождение суммы чисел, расположенных по адресам* $\alpha_1, \ldots, \alpha_n$ *(числа-адреса* $\alpha_1, \ldots, \alpha_n$ *не образуют закономерного ряда).*

Пусть

$$'\,(\alpha \oplus i) = \alpha_i;\quad '\varphi = \alpha \oplus 1;\quad '\varphi 1 = \alpha \oplus n$$

и результат помещается по адресу β. При этом получим

$$
\begin{array}{c}
0 \to \beta \\
K\ \ldots\ '\beta + {}^3\varphi \to \beta \\
'\varphi \oplus 1 \to \varphi \\
\boldsymbol{P}\,\{'\varphi \leqslant '\varphi 1\}\,K \downarrow \mathcal{Я}.
\end{array}
$$

4. СТАНДАРТНЫЕ СПОСОБЫ ЗАДАНИЯ ИНФОРМАЦИИ

Для наиболее употребительных операций следования (например, для операции $\bigoplus 1$) можно принять стандартный способ кодирования информации, при котором вся информация о последовательности передается программе с помощью одного адреса. Для этого в последовательности элементов, заданной адресами

$$\alpha \bigoplus 1, \quad \alpha \bigoplus 2, \quad \ldots, \quad \alpha \bigoplus n,$$

удобно положить $'\alpha = n$, где n — число элементов в последовательности.

Адрес α, содержащий информацию о размере массива, назовем ведущим адресом α-последовательности. Вся информация об α-последовательности может быть передана программе путем засылки ее ведущего адреса по некоторому адресу f_0, по которому программой осуществляется обозревание всей последовательности.

Условимся α — последовательность адресов обозначать через $\overline{\alpha}$, а последовательность ее содержимых через $' \overline{\alpha}$.

Рассмотрим примеры.

Пример 1. Положим в примере 5 предыдущего параграфа, что исходная информация задана в виде α-последовательности, т. е. $'\alpha = n$; $'(\alpha \bigoplus i) = a_i$ $(i = 1, 2, \ldots, n)$.

Принимая, что $'f_0 = \alpha$, получаем

$$0 \to \beta; \quad {}'f_0 \to \varphi$$
$$K \ \ldots \ {}'\varphi \bigoplus 1 \to \varphi$$
$$P\,\{{}''\varphi \in \overline{M}\}\,\mathcal{A}$$
$$P\,\{{}'\varphi \leqslant {}'f_0 \bigoplus {}''f_0\} \downarrow \mathcal{A}$$
$$\beta + {}^2\varphi \to \beta; \quad K.$$

Пример 2. Зададим слова (в некотором алфавите) в виде α-последовательностей так, что по адресу α помещается длина слова (число букв), по адресу $\alpha + s$ — его s-ая буква.

Требуется в слово Y, заданное ω-последовательностью, после его p-й буквы вставить слово X, заданное ν-последовательностью.

Исходное адресное отображение

ω	$\omega \bigoplus 1$		$\omega \bigoplus p$		$\omega \bigoplus m$
m	y_1	$\ldots$	y_p	$\ldots$	y_m
ν	$\nu \bigoplus 1$				$\nu \bigoplus s$
s	x_1		$\ldots$		x_s

Результативное адресное отображение

ω	ω⊕1	ω⊕p	ω⊕p⊕1	. . .	ω⊕p⊕s	ω⊕p⊕s⊕1		ω⊕m⊕s
m⊕s	y_1	. . .	y_p	x_1		x_s	y_{p+1}	. . .	y_m

Число p предполагается заданным как содержимое адреса $'i$, причем $1 \leqslant 'i \leqslant m$.

Составим подпрограмму для реализации этого алгоритма с формулой вхождения $П\{\nu, \omega, i\}$.

Вставка . . . $\varnothing \to \nu$, $\varnothing \to \omega$, $\varnothing \to i$

$Ц\{'\omega(1)'i \oplus 1 \to \pi\}$

$'(\omega \oplus \pi) \to \omega \oplus \pi \oplus '\nu$

$Ц\{1(1)'\nu \to \pi\}$

$'(\nu \oplus \pi) \to \omega \oplus 'i \oplus \pi$

$'\omega \oplus '\nu \to \omega$

$Я$.

Первый цикл осуществляет сдвиг букв слова Y от буквы y_{p+1} по букву y_m (вправо по последовательности адресов; при этом вначале сдвигается первая справа буква слова, затем предшествующая ей и т. д.). При переписывании букв в естественном порядке их записи при $s < m - p$ исказился бы остаток слова Y.

В заключение приведем несколько более сложную схему обозревания последовательности.

Пример 3. Рассмотрим алгоритм формальной проверки правильности записи формул (арифметических выражений) по Лукасевичу [25, 26].

При любом способе записи формул нужно однозначно указывать для каждой операции ее область действия. В записи фурмул со скобками область действия операций указывается при помощи скобок и соглашения о старшинстве операций. При этом, если в формулах ограничиться указанием областей действия операций только таким образом, количество элементов информации может значительно возрасти. Лукасевич показал, что скобки в формулах не составляют необходимую информацию и могут быть исключены, если принять специальный способ записи формул: ставить знаки одноместных и двухместных операций перед аргументом. Например, скобочная запись формулы

$$b + \cos(a + b) \times c$$

в записи Лукасевича принимает вид

$$+ b \times \cos + abc.$$

Форма Лукасевича также может использоваться для записи высказываний, заданных на языке двухпозиционной математической логики.

Таким образом, в бесскобочной записи формул участвуют следующие элементы: величины, одноместные (унарные) и двухместные (бинарные) операции.

Предположим, что при кодировании формулы каждый ее элемент снабжается специальным признаком:

a — признак величины;

O_1 — признак одноместной операции;

O_2 — признак двухместной операции.

При формальной проверке правильности записи формулы играют роль лишь эти признаки. Отождествим каждую величину, одноместную или двухместную операцию с ее признаком и в этих терминах сформулируем алгоритм проверки, т. е. будем рассматривать слова в алфавите $\{a, O_1, O_2\}$, предполагая их побуквенно закодированными в последовательность.

Можно показать, что необходимое и достаточное условие того, чтобы слово в алфавите $\{a, O_1, O_2\}$ являлось формулой по Лукасевичу, состоит в следующем:

1) число вхождений в слово буквы a на одно больше числа вхождений буквы O_2;

2) часть слова, следующая за каждой буквой O_1, представляет собой одну или несколько последовательно записанных формул;

3) часть слова, следующая за каждой буквой O_2, представляет собой две или более последовательно записанные формулы.

Воспользуемся сформулированным условием для построения алгоритма формальной проверки правильности записи формул по Лукасевичу.

Алгоритм будет осуществлять свою работу при однократном поэлементном (побуквенном) обозревании информации, начиная с последнего элемента и кончая первым. Представленная запись может кроме символов величин и операций содержать «несвойственные» символы b, появившиеся в результате ошибок.

В результате правильного задания формулы по адресу γ алгоритмом засылается символ 1, а в противном случае, символ 0. В последнем случае при появлении в записи несвойственного элемента его адрес содержится по адресу φ.

Для построения алгоритма введем счетчик s числа формул, предшествующих обозреваемому знаку операции. Вначале $'s = 0$; при обозревании величины в счетчик прибавляется единица, при появлении знака двухместной операции вычитается единица, а при появлении одноместной операции содержимое адреса не изменяется; тогда признаком правильности записи формулы является:

1) перед знаком одноместной операции $'s \geqslant 1$;

2) перед знаком двухместной операции $'s \geqslant 2$;

3) в конце проверки формулы $s = 1$.

Исходное адресное отображение

$'f_0 = \alpha$

$'\alpha = n$

$'(\alpha \oplus 1)$ ⎫
⋮ ⎬ поэлементно закодированная исходная формула
⋮ ⎪
$'(\alpha \oplus 'n)$ ⎭

Результативное отображение

$'\gamma \geqslant \alpha$;

$'\gamma$ ($'\gamma = 1$, если входное слово — формула;

Программа

$0 \to s$; $0 \to \gamma$

M_1 ... $\boldsymbol{Ц}\{''f_0 (\ominus 1)\, 1 \to \delta\}\ M_4$

$\boldsymbol{P}\{'('f_0 \oplus 'n) = O_1\}\ M_2$

$\boldsymbol{P}\{'('f_0 \oplus 'n) = O_2\}\ M_3$

$\boldsymbol{P}\{'('f_0 \oplus 'n) = a\}\ 's + 1 \to s;\ M_1 \downarrow \mathcal{B}$

M_2 ... $\boldsymbol{P}\{'s \geqslant 1\}\ M_1 \downarrow \mathcal{B}$

$'\gamma = 0$, если входное слово не является формулой);
$'\varphi$ ($'\varphi$ равно номеру несвойственного элемента)

M_3 ... $\boldsymbol{P}\{'s \geqslant 2\}\ 's - 1 \to s;\ M_1 \downarrow \mathcal{A}$
M_4 ... $\boldsymbol{P}\{'s = 1\}\ 1 \to \gamma$
$\mathcal{A}$

Приведенный алгоритм может быть записан в более компактной форме, если воспользоваться следующим соглашением Предположим, что коды букв a, O_1, O_2, b принадлежат множеству адресов; поместим по этим адресам метки алгоритма

$$\begin{aligned} 'O_1 &= M_2;\\ 'O_2 &= M_3;\\ 'a &= M_5:\\ 'b &= \mathcal{A}, \end{aligned} \tag{3.28}$$

тогда приведенный алгоритм можно записать в более компактной форме:

$$0 \to s;\ \ 0 \to \gamma$$
$$M_1 \ \ldots\ \boldsymbol{Ц}\{''f_0 (\ominus 1)\ 1 \to \delta\}\ M_4$$
$$''('f_0 \oplus 'b)$$
$$M_2 \ \ldots\ \boldsymbol{P}\{'s \geqslant 1\}\ M_1 \downarrow \mathcal{A}$$
$$M_3 \ \ldots\ \boldsymbol{P}\{'s \geqslant 2\}\ 's - 1 \to s;\ \ M_1 \downarrow \mathcal{A}$$
$$M_5 \ \ldots\ 's + 1 \to s;\ \ M_1$$
$$M_4 \ \ldots\ \boldsymbol{P}\{'s = 1\}\ 1 \to \gamma$$
$$\mathcal{A}$$

Здесь третья строка представляет собою формулу безусловного перехода. Если адреса O_1, O_2, a, b оказались бы использованными для каких-либо целей, то вместо них для кодирования соответствующих меток могла бы быть выбрана какая-либо простейшая функция от этих кодов, например функция вида

$$k \oplus \theta,$$

где θ — один из указанных кодов, а k подобрано так, чтобы значения этой функции при $\theta = O_1$, O_2, a, b были бы свободными адресами. Соответственно этому в исходном отображении вместо содержимых адресов O_1, O_2, a, b следовало бы задать:

$$\begin{aligned} '(k \oplus O_1) &= M_2;\\ '(k \oplus O_2) &= M_3;\\ '(k \oplus a) &= M_5;\\ '(k \oplus b) &= \mathcal{A}, \end{aligned} \tag{3.29}$$

а формулу безусловного перехода $''('f_0 \oplus 'b)$ заменить формулой $'(k \oplus '('f_0 \oplus 'b))$.

Совокупность соотношений (3.28) или (3.29) назовем щитом переключателя, а соответствующую формулу безусловного перехода — переключателем.

В адресном языке возможна запись аналогичных алгоритмов без соответствующих щитов переключателей, если вместо меток M_2,

M_2, M_3 применить непосредственно символы (их коды) O_1, O_2, a, b. В этом случае приведенная программа запишется в виде

$$
\begin{gathered}
0 \to s, \quad 0 \to \gamma \\
M_1 \ \ldots \ \boldsymbol{Ц}\{{}'f_0 (\ominus 1)\, 1 \to \delta\}\, M_4 \\
{}'({}'f_0 \oplus {}'\delta) \\
O_1 \ \ldots \ \boldsymbol{Р}\{{}'s \geqslant 1\}\, M_1 \downarrow \mathcal{Я} \\
O_2 \ \ldots \ \boldsymbol{Р}\{{}'s \geqslant 2\}{}'s - 1 \to s; \quad M_1 \downarrow \mathcal{Я} \\
a \ \ldots \ {}'s + 1 \to s; \quad M_1 \\
M_4 \ \ldots \ \boldsymbol{Р}\{{}'s = 1\}\, 1 \to \gamma \\
b \ \ldots \ \mathcal{Я}.
\end{gathered}
$$

5. СХЕМЫ ОДНОВРЕМЕННОГО ОБОЗРЕВАНИЯ РЯДА ПОСЛЕДОВАТЕЛЬНОСТЕЙ

При рассмотрении нескольких последовательностей считаем, что в каждой из них задана некоторая операция следования. Однако разнообразие схем обозревания увеличивается.

Если операции следования применяются в схеме одновременно во всех последовательностях, то программа представляет собой простой циклический процесс. Такие схемы рассматриваются в этом параграфе.

Пусть задан ряд последовательностей кодов

$$
\left.
\begin{array}{l}
a_{11}, \ a_{12}, \ \ldots \ a_{1n}; \\
a_{21}, \ a_{22}, \ \ldots, \ a_{2n}; \\
\ldots\ldots\ldots \\
a_{k1}, \ a_{k2}, \ \ldots, \ a_{kn},
\end{array}
\right\} \tag{3.30}
$$

расположенных соответственно по адресам

$$
\left.
\begin{array}{l}
\alpha_{11}, \ \alpha_{12}, \ \ldots, \ \alpha_{1n}; \\
\alpha_{21}, \ \alpha_{22}, \ \ldots, \ \alpha_{2n}; \\
\ldots\ldots\ldots \\
\alpha_{k1}, \ \alpha_{k2}, \ \ldots, \ \alpha_{kn},
\end{array}
\right\} \tag{3.31}
$$

т. е.

$${}'\alpha_{ij} = a_{ij} \ (i = 1, \ 2, \ \ldots, \ k; \quad j = 1, \ 2, \ \ldots, \ n).$$

Пусть в каждой из последовательностей адресов (3.31) определены операции следования C_i $(i = 1, \ 2, \ \ldots, \ k)$, которые в программе применяются одновременно во всех последовательностях. Такую программу назовем схемой одновременного обозревания k последовательностей.

Введем адреса $\varphi_i (i = 1, 2, \ldots, k)$ для первых кодов последовательностей (3.30):

$$^2\varphi_i = a_{i1}; \quad '\varphi_i = \alpha_{i1} \ (i = 1, 2, \ldots, k)$$

и пусть $'\overline{\varphi}_j = \alpha_{jn}$ для некоторого $j (1 \leqslant j \leqslant k)$.

Тогда схему одновременного обозревания кодов (3.30) можно представить в виде

$$\begin{gathered} B \ \ldots \ \boldsymbol{P}\{'\varphi_j \leqslant '\overline{\varphi}_j\} \downarrow \mathcal{A} \\ \Phi\{^2\varphi_1, \ \ldots \ ^2\varphi_k\} \\ C_1{}'\varphi_1 \Rightarrow \varphi_1, \ \ldots, \ C_k{}'\varphi_k \Rightarrow \varphi_k \\ B, \end{gathered} \tag{3.32}$$

или с формулой циклирования

$$\begin{gathered} B \ \ldots \ \boldsymbol{Ц}\{'\varphi_1, \ C_1 \varnothing, \Rightarrow \varphi_1; \ \ldots \ ; \ '\varphi_j, \ C_j \varnothing, \ '\varphi 1_j \Rightarrow \varphi_j; \ \ldots \ ; \\ '\varphi_k, \ C_k \varnothing \Rightarrow \varphi_k\}, \ \mathcal{A} \\ \Phi\{^2\varphi_1, \ \ldots, \ ^2\varphi_k\} \\ C_1{}'\varphi_1 \Rightarrow \varphi_1, \ \ldots, \ C_k{}'\varphi_k \Rightarrow \varphi_k \\ B. \end{gathered}$$

Все рассмотренные типы схем обозревания последовательности могут быть непосредственно применены к случаю одновременного обозревания ряда последовательностей.

В частном случае, когда операции следования во всех последовательностях одинаковы, т. е. $C_1 = C_2 = \ldots = C_k = C$, схему одновременного обозревания и операцию следования последовательностей назовем групповыми.

Для групповой операции следования вида

$$\alpha_{ij+1} = \alpha_{ij} \oplus 1$$

схема с адресами сдвига несколько упрощается (вначале $'\beta = 0$, $'\beta 1 = n$):

$$\begin{gathered} B \ \ldots \ '\beta \oplus 1 \Rightarrow \beta \\ \boldsymbol{P}\{'\beta \leqslant '\beta 1\} \downarrow \mathcal{A} \\ \Phi('('\varphi_1 \oplus '\beta); \ '('\varphi_2 \oplus '\beta); \ \ldots \ ; \ '('\varphi_n \oplus '\beta)) \\ B. \end{gathered} \tag{3.33}$$

Приведем запись той же схемы с применением формулы циклирования:

$$\begin{gathered} \boldsymbol{Ц}\{1\,(1)'\beta 1 \Rightarrow \beta\} \\ \Phi('('\varphi_1 \oplus '\beta); \ '('\varphi_2 \oplus '\beta); \ \ldots \ ; \ '('\varphi_n \oplus '\beta)) \\ \mathcal{A}. \end{gathered}$$

Предлагаем читателю проделать то же самое для всех других схем.

Рассмотрим ряд примеров.

Пример 1. *Групповой перенос информации.*

А. Элементы информации, помещенные в последовательности адресов

$$\alpha,\ \alpha \oplus d,\ \alpha \oplus 2d,\ \ldots,\ \alpha \oplus nd, \tag{3.34}$$

перенести соответственно в последовательность адресов

$$\beta,\ \beta \oplus p,\ \ldots,\ \beta \oplus np. \tag{3.35}$$

Предполагается, что последовательности адресов не налагаются одна на другую.

Пусть $'\varphi = \alpha$; $'\psi = \beta$. При этом получаем программу

$$\begin{gathered} B\ \ldots\ \boldsymbol{P}\,\{'\varphi \leqslant \alpha \oplus nd\} \downarrow \mathcal{A} \\ {}^{2}\varphi \to {}'\psi \\ {}'\varphi \oplus d \to \varphi;\ {}'\psi \oplus p \to \psi \\ B, \end{gathered}$$

или

$$\begin{gathered} \boldsymbol{Ц}\,\{'\varphi\,(d)\ \alpha \oplus nd \to \varphi;\ {}'\psi\,(p) \to \psi\} \\ {}^{2}\varphi \to {}'\psi \\ \mathcal{A} \end{gathered}$$

(последнее содержимое адреса ψ не указано, так как оно определяется последним содержимым адреса φ).

Эти программы не зависят явно от начального адреса последовательности адресов (3.35); однако они могут быть преобразованы в программы, которые не будут зависеть явно также и от начального адреса последовательности (3.34), от количества рассматриваемых элементов и от шагов d и p в них.

Пусть $'\varphi 1 = \alpha + nd$; $'\delta_1 = d$; $'\delta_2 = p$; получаем программу

$$\begin{gathered} B\ \ldots\ \boldsymbol{P}\,\{'\varphi \leqslant {}'\varphi 1\} \downarrow \mathcal{A} \\ {}^{2}\varphi \to {}'\psi \\ {}'\varphi \oplus {}'\delta_1 \to \varphi;\ {}'\psi \oplus {}'\delta_2 \to \psi \\ \mathcal{A}. \end{gathered}$$

Таким образом, эта программа реализует групповой перенос из произвольной последовательности адресов (3.34) в произвольную (не пересекающуюся с ней) последовательность адресов (3.35) и имеет стандартный вид: число элементов учитывается содержимым $\varphi 1$, начальные номера соответствующих последовательностей адресов — содержимым адресов φ и ψ, шаги последовательностей адресов — содержимым адресов δ_1 и δ_2.

Б. Построить программу для предыдущей задачи, используя адреса сдвига. Примем введенные ранее обозначения и, кроме того, положим: $'\gamma_1 = 0$; $'\gamma_2 = 0$. Тогда получим программу

$$\begin{gathered} B\ \ldots\ \boldsymbol{P}\,\{'\gamma_1 \leqslant nd\} \downarrow \mathcal{A} \\ {}'('\varphi \oplus {}'\gamma_1) \to {}'\psi \oplus {}'\gamma_2 \\ {}'\gamma_1 \oplus {}'\delta_1 \to \gamma_1;\ {}'\gamma_2 \oplus {}'\delta_2 \to \gamma_2 \\ B. \end{gathered}$$

В. Элементы информации, расположенные по адресам α, $\alpha + 1$, ..., $\alpha + n$, перенести в последовательность адресов β_0, β_1, β_2, ..., β_n, где числа β_i не образуют арифметической прогрессии (перенос информации по адресам высших рангов). Вводим адреса

$$
\begin{array}{ll}
'\beta = \beta_0; & '\psi = \beta; \\
'(\beta \oplus 1) = \beta_1; & '\varphi = \alpha; \\
\ldots\ldots & '\varphi 1 = \alpha \oplus n. \\
'(\beta \oplus n) = \beta_n; &
\end{array}
$$

Программа переноса в этом случае может быть представлена следующим образом:

Результат	$B \ldots {}^2\varphi \to {}^2\psi$
'(β-последовательность)	$'\varphi \oplus 1 \to \varphi;\ '\psi \oplus 1 \to \psi$
	$\boldsymbol{P}\{'\varphi \leqslant '\varphi 1\}\ B \downarrow \mathcal{A}.$

Программа может быть записана в другом виде, если ввести адрес сдвига δ ($'\delta = 0$):

$$
\begin{gathered}
B \ldots {}'('\varphi \oplus '\delta) \to '('\psi \oplus '\delta) \\
'\delta \oplus 1 \to \delta \\
\boldsymbol{P}\{'\delta \leqslant '\bar{\delta}\}\ B \downarrow \mathcal{A},
\end{gathered}
$$

где $'\bar{\delta} = n$.

Пример 2. *Вычисление скалярного произведения двух векторов.* Пусть компоненты векторов $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_n$ размещены соответственно по адресам $\alpha \oplus 1$, $\alpha \oplus 2$, ..., $\alpha \oplus n$ и $\beta \oplus 1$ $\beta \oplus 2$, ..., $\beta \oplus n$; получаемое произведение помещается по адресу γ. Кроме того, положим:

$$'\varphi = \alpha \oplus 1;\ '\psi = \beta \oplus 1;\ '\bar{\varphi} = \alpha \oplus n.$$

Тогда получим программу

$$
\begin{gathered}
0 \to \gamma \\
K \ldots {}^2\varphi \times {}^2\psi + '\gamma \to \gamma \\
'\varphi \oplus 1 \to \varphi;\ '\psi \oplus 1 \to \psi \\
\boldsymbol{P}\{'\varphi \leqslant '\bar{\varphi}\}\ K \downarrow \mathcal{A}.
\end{gathered}
$$

В частном случае, когда $\alpha \oplus m = \beta$, программа имеет вид

$$
\begin{gathered}
0 \to \gamma \\
K \ldots {}^2\varphi \times '('\varphi \oplus m) + '\gamma \to \gamma \\
'\varphi \oplus 1 \to \varphi \\
\boldsymbol{P}\{'\varphi \leqslant '\bar{\varphi}\}\ K \downarrow \mathcal{A}.
\end{gathered}
$$

Вариант программы с адресом сдвига и со счетчиком, работающим по принципу вычитания, предлагаем рассмотреть читателю самостоятельно.

Пример 3. *Вычисление и запоминание значений функции от ряда значений векторного аргумента.* Пусть требуется вычислить значение функции f от n переменных $x_1, x_2, \ldots, x_n$ для ряда значений ее аргументов:

$$x_{11}, x_{12}, \ldots, x_{1n}$$
$$x_{21}, x_{22}, \ldots, x_{2n}$$
$$\ldots\ldots\ldots$$
$$x_{k1}, x_{k2}, \ldots, x_{kn}.$$

Пусть $'(\alpha_i \oplus j) = x_{ij}$ $(1 \leqslant i \leqslant n;\ \ 1 \leqslant j \leqslant k)$ и вычисленные значения функции f запоминаются в адресах $\beta \oplus 1, \beta \oplus 2, \ldots, \beta \oplus k$ и

$$'\gamma_i = \alpha_i \oplus 1 \ (1 \leqslant i \leqslant n);\quad '\varphi = \beta \oplus 1;\quad '\varphi 1 = \beta \oplus k.$$

В результате получаем программу

$$B \ldots f(^2\gamma_1, {}^2\gamma_2, \ldots, {}^2\gamma_n) \to '\varphi$$
$$'\gamma_1 \oplus 1 \to \gamma_1;\ \ldots;\ '\gamma_n \oplus 1 \to \gamma_n;\ '\varphi \oplus 1 \to \varphi$$
$$\boldsymbol{P}\{'\varphi \leqslant '\varphi 1\}\ B \downarrow \mathcal{Я}.$$

Если $\alpha_{i+1} = \alpha_i \oplus k$; $\alpha_i = \alpha\,(i = 1, 2, \ldots, n-1)$, то можно ограничиться одним адресом $'\gamma = \alpha \oplus 1$. При этом программа принимает вид:

$$B \ldots f(^2\gamma;\ '('\gamma \oplus k);\ \ldots;\ '('\gamma \oplus k\,(n \ominus 1))) \to '\varphi$$
$$'\gamma \oplus 1 \to \gamma;\ '\varphi \oplus 1 \to \varphi$$
$$\boldsymbol{P}\{'\varphi \leqslant '\varphi 1\}\ B \downarrow \mathcal{Я}.$$

Рекомендуем читателю самостоятельно записать программы для приведенных примеров с применением формулы циклирования.

Пример 4. *Вычисление m-ой степени подстановки* $i_1, i_2, \ldots, i_k$ чисел $1, 2, \ldots, k$[1].

Пусть исходная подстановка поэлементно закодирована в виде α-последовательности, а ее m-я степень должна быть получена в виде β-последовательности (последовательности α и β — не пересекающиеся) и, кроме того, $'f_0 = \alpha$, $'\psi_0 = \beta$.

При этом искомый алгоритм может быть представлен в виде

$$M \ldots '(\ldots{}'('('f_0 \oplus 'i) \oplus 'f_0) \oplus \ldots \oplus 'f_0) \to '\psi_0 \oplus 'i$$
$$'i \oplus 1 \to i$$
$$\boldsymbol{P}\{'i \leqslant ''f_0\}\ M \downarrow \mathcal{Я},$$

или с использованием формулы $\boldsymbol{Ц}$

$$\boldsymbol{Ц}\{(m) \to i\}\ M$$
$$'('f_0 \oplus i) \to '\psi_0 \oplus 'i$$
$$\boldsymbol{Ц}\{(k) \to l\}$$
$$'('('\psi_0 \oplus 'i) \oplus 'f) \to '\psi_0 \oplus 'i$$
$$M \ldots \mathcal{Я}.$$

---

[1] Этот пример принадлежит А. А. Летичевскому.

6. ОБЩИЕ СХЕМЫ ОБОЗРЕВАНИЯ РЯДА ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Пусть задан ряд последовательностей кодов

$$\begin{array}{l} a_{11},\ a_{12},\ \ldots,\ a_{1n_1}; \\ a_{21},\ a_{22},\ \ldots,\ a_{2n_2}; \\ \ldots\ldots\ldots\ldots \\ a_{k1},\ a_{k2},\ \ldots,\ a_{kn_k}, \end{array} \tag{3.36}$$

расположенных по адресам

$$\begin{array}{l} \alpha_{11},\ \alpha_{12},\ \ldots,\ \alpha_{1n_1}; \\ \alpha_{21},\ \alpha_{22},\ \ldots,\ \alpha_{2n_2}; \\ \ldots\ldots\ldots\ldots \\ \alpha_{k1},\ \alpha_{k2},\ \ldots,\ \alpha_{kn_k}, \end{array} \tag{3.37}$$

т. е.

$$\alpha_{ij} = a_{ij}\ (i = 1,\ 2,\ \ldots\ k;\ j = 1,\ 2,\ \ldots,\ n_i).$$

Пусть в каждой из последовательностей адресов (3.37) определена операция следования C_i $(i = 1,\ 2,\ \ldots,\ k)$. В общем случае порядок применения операций следования в программах может зависеть от некоторых условий. Это приводит к сложным циклическим процессам.

Проиллюстрируем способы построения адресных программ с общими схемами обозревания ряда последовательностей на нескольких примерах.

Пример 1. *Вычисление значений функции f от трех аргументов a, b, c, каждый из которых задан набором значений, размещенных соответственно по адресам*

$$\begin{array}{l} \alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha \oplus n; \\ \beta \oplus 1,\ \beta \oplus 2,\ \ldots,\ \beta \oplus m; \\ \gamma \oplus 1,\ \gamma \oplus 2,\ \ldots,\ \gamma \oplus p, \end{array}$$

для всевозможных их сочетаний с выдачей результатов на печать.

Пусть имеется подпрограмма (с меткой f) вычисления по заданным значениям аргументов a, b, c значений функции f. Формула вхождения на эту подпрограмму пусть имеет вид $\Pi f\ \{a,\ b,\ c, \delta\}$ и означает, что если a, b, c — аргументы функции, то по адресу δ будет получено соответствующее значение $f(a,\ b,\ c)$. Искомая программа может быть записана в виде

$$\begin{array}{c} \Ц\ \{1\ (1)\ n \Rightarrow \pi_1\},\ \mathcal{G} \\ \Ц\ \{1\ (1)\ m \Rightarrow \pi_2\} \\ \Ц\ \{1\ (1)\ p \Rightarrow \pi_3\}\ Z \\ \Pi f\ \{'(\alpha \oplus '\pi_1),\ '(\beta \oplus '\pi_2),\ '(\gamma \oplus '\pi_3);\ d\} \\ \text{Печать}\ 'd \\ Z\ldots \end{array}$$

Формулы циклирования могут быть переставлены местами, в связи с чем изменится порядок выдачи результатов на печать.

Пример 2. *Перенос элементов, обладающих свойством* μ. Требуется элементы α-последовательности, обладающие данным свойством μ, в порядке их следования перенести в β-последовательность.

Обозначим через $L(\mu, a)$ условие наличия свойства μ у элемента a. Пусть адреса φ и ψ фиксируют начала рассматриваемых последовательностей, т. е. $'\varphi = \alpha + 1$; $'\psi = \beta$ и $'\varphi 1 = \alpha \oplus n$, где n — число членов в α-последовательности. Тогда программа может быть записана в виде

$$
\begin{gathered}
B \ldots \boldsymbol{P}\{\mu, {}^{2}\varphi)\} \downarrow k \\
{}'\psi \oplus 1 \to \psi \\
{}^{2}\varphi \to {}'\psi \\
k \ldots {}'\varphi \oplus 1 \to \varphi \\
\boldsymbol{P}\{{}'\varphi \leqslant {}'\varphi 1\}\ B \downarrow \mathcal{B}.
\end{gathered}
$$

В отличие от программ, рассмотренных в предыдущем параграфе, формулы во второй и четвертой строках данной программы не перестановочны. В конце работы алгоритма адрес ψ содержит число $\beta \oplus m$, где m — число перенесенных элементов (если в α-последовательности нет элементов, обладающих свойством μ, $'\psi = \beta$).

По первой предикатной формуле проверяется наличие свойства μ у элемента, отмеченного фиксатором φ; вторая строка — сдвиг фиксатора ψ — означает подготовку места для переноса обнаруженного элемента со свойством μ; третья строка осуществляет этот перенос; строка с меткой k — сдвиг фиксатора φ — является переходом к обозреванию следующего элемента в α-последовательности; последняя строка — проверка конца работы алгоритма.

Рассмотрим другой вариант приведенной программы.

Введем адреса сдвига i и β, $'\alpha = n$ и по адресу β поместим число m. Тогда приведенная программа может быть записана в виде

$$
\begin{gathered}
B \ldots 1 \to i;\ 0 \to \beta \\
k1 \ldots \boldsymbol{P}\{L(\mu', (\alpha \oplus {}'i)) \downarrow k \\
{}'\beta \oplus 1 \to \beta \\
{}'(\alpha \oplus {}'i) \to \beta + {}'\beta \\
k \ldots {}'i \oplus 1 \to i \\
\boldsymbol{P}\{{}'i \leqslant {}'\alpha\}\ k1 \downarrow \mathcal{B}
\end{gathered}
$$

или с формулой циклирования в виде

$$
\begin{gathered}
0 \to \beta \\
\boldsymbol{Ц}\{1\ (1)'\alpha \to i\}\ k,\ \mathcal{B} \\
P\{L(\mu,\ {}'(\alpha \oplus {}'i))\} \downarrow k1 \\
{}'\beta \oplus 1 \to \beta \\
{}'(\alpha \oplus {}'i) \to \beta \oplus {}'\beta \\
k1 \ldots k \ldots
\end{gathered}
$$

Пример 3. *Выполнение операций над многочленами.* Для кодирования на ЭВЦМ многочленов, например, от трех переменных

$$A = \sum_{s=1}^{k} a_{i_s j_s k_s} x^{i_s} y^{j_s} z^{k_s}$$

(где i_s, j_s, k_s — некоторые целые) можно поступить так.

Для каждого многочлена с m членами можно выделить последовательность $2m$ ячеек

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha + 2m.$$

При этом для запоминания каждого из членов отводятся две соседние ячейки, в первую из которых заносятся показатели при x, y, z, а во вторую — коэффициенты. Однако для удобства в дальнейшем считаем, что показатели степени и коэффициенты каждого члена размещены в определенных разрядах одного и того же адреса.

А. Сжатие информации. В результате выполнения тех или иных операций над многочленами могут появиться члены, равные нулю. Для очистки адресов, занятых нулевыми членами, введем алгоритм «сжатия» информации о многочлене. Сжатая информация может помещаться по адресам, в которых находился до «сжатия» данный многочлен, или в другую последовательность адресов.

Алгоритм сжатия многочлена сводится к предыдущему, если в качестве μ рассматривать свойство «коэффициент члена отличен от нуля».

Б. Приведение подобных членов многочлена. Требуется в многочлене A привести подобные члены и полученный многочлен B поместить в β-последовательность, а по адресу β поместить число m, равное числу членов в B. При этом допускается равенство $\alpha = \beta$, т. е. приведенный многочлен может размещаться в той же α-последовательности.

Примем следующий порядок работы:

1) фиксируем первый ненулевой член A; среди оставшихся разыскиваем ему подобные, которые к нему приводим, т. е. их коэффициенты прибавляем к зафиксированному, а в адреса, в которых они хранились, засылаем нули;

2) если полученный приведенный член отличен от нуля, то его переносим в β-последовательность;

3) операции п. 1 применяем к следующему ненулевому члену A и т. д.;

4) вычисления прекращаем после фиксации последнего члена A.

Обозначим через a_n и a_k подадреса показателей и коэффициента члена, содержащегося по адресу a. Рассмотрим вариант программы с адресами сдвига.

Пусть:

$'\varphi = \alpha$; $'\xi = \beta$;

i — адрес сдвига по последовательности α, отмечает номер фиксируемого (ненулевого) ее элемента;

β — адрес сдвига по последовательности β, отмечает номер отличного от нуля приведенного члена B;

ψ — адрес сдвига по последовательности α, отмечает номера членов A, подобных члену, номер которого содержится по i.

В принятых обозначениях адресная программа приведения подобных членов многочлена может быть записана в виде

$$0 \to i;\quad {}''\varphi \to \alpha 1;\quad 0 \to {}'\xi$$
$$k0\ \ldots\ {}'i \oplus 1 \to i$$
$$P\{i \leqslant {}'\alpha 1\} \downarrow \mathcal{A}$$
$$P\{{}'({}'\varphi \oplus' i)_k \neq 0\} \downarrow k0$$
$${}'i \oplus 1 \to \psi$$
$$M\ \ldots\ P\{{}'\psi \leqslant {}'\alpha 1\} \downarrow S$$
$$P\{{}'({}'\varphi \oplus {}'i)_n = {}'({}'\varphi \oplus {}'\psi)_n\} \downarrow k1$$
$${}'({}'\varphi \oplus {}'i)_k + {}'({}'\varphi \oplus {}'\psi)_k \to ({}'\varphi \oplus {}'i)_k$$
$$0 \to {}'\varphi \oplus {}'\psi$$
$$k1\ \ldots\ {}'\psi \oplus 1 \to \psi$$
$$M$$
$$S\ \ldots\ P\{{}'({}'\varphi \oplus {}'i)_k \neq 0\}''\xi \oplus 1 \to {}'\xi;\ {}'({}'\varphi \oplus {}'i) \to {}'\xi \oplus {}''\xi \downarrow$$
$$k0,$$

или с формулой циклирования в виде

$${}''\varphi \to \alpha 1;\ 0 \to {}'\xi$$
$$Ц\{1\ (1)\ {}'\alpha 1 \to i\}\ k0,\ \mathcal{A}$$
$$P\{{}'({}'\varphi \oplus {}'i)_k \neq 0\} \downarrow k0$$
$$Ц\{{}'i \oplus 1\ (1)\ {}'\alpha 1 \to \psi\}\ k1$$
$$P\{{}'({}'\varphi \oplus {}'i)_n = {}'({}'\varphi \oplus {}'\psi_n)\} \downarrow k1$$
$${}'({}'\varphi \oplus {}'i)_k + {}'({}'\varphi \oplus {}'\psi)_k \to ({}'\varphi \oplus {}'i)_k$$
$$0 \to {}'\varphi \oplus {}'\psi$$
$$k1\ \ldots\ P\{{}'({}'\varphi \oplus {}'i)_k \neq 0\}\ {}''\xi \oplus 1 \to {}'\xi;\ {}'({}'\varphi \oplus {}'i) \to {}'\xi \oplus {}''\xi \downarrow$$
$$k0\ldots$$

7. СХЕМЫ ПРОСМОТРА МАССИВОВ

При решении многих задач данные часто размещаются в виде двумерных и многомерных таблиц или других массивов, обозревание которых может осуществляться по нескольким операциям следования. Так, например, в матрице с элементами a_{ij} $(1 \leqslant i \leqslant n;\ 1 \leqslant j \leqslant m)$ можно рассматривать две операции следования, первая из которых C_1 по данному элементу a_{ij} позволяет определить следующий элемент той же строки

$$Ca_{ij} = a_{ij+1},$$

а вторая C_2 — следующий элемент $a_{i+1,\,j}$ того же столбца.

Рассмотрим примеры:

Пример 1. *Вычисление квадрата квадратной матрицы.* Пусть элементы матрицы A порядка n размещены строками по адресам

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha \oplus n^2$$

и элементы C_{ij} получаемой матрицы A^2 размещаются также строками по адресам

$$\beta \oplus 1,\ \beta \oplus 2,\ \ldots,\ \beta \oplus n^2.$$

Примем обычный порядок работы: скалярное произведение k-й строки на j-й столбец образует элемент c_{kj} матрицы A^2:

$$\sum_{i=1}^{n} a_{ki} \cdot a_{ij} = c_{kj}.$$

Обозначая адреса для хранения индексов переменных k, j, i теми же буквами k, j, i и применяя формулу циклирования **Ц**, адресную программу вычисления квадрата квадратной матрицы можно представить в виде

$$\begin{gathered}
\textbf{Ц}\,\{1\,(1)\,n \to k\} \\
\textbf{Ц}\,\{1\,(1)\,n \to j\} \\
\textbf{Ц}\,\{1\,(1)\,n \to i\} \\
'(\alpha \oplus ('k \ominus 1)\,n \oplus 'i) \times '(\alpha \oplus ('i \ominus 1)\,n \oplus 'j) + \\
+\ '(\beta \oplus ('k \ominus 1)\,n \oplus 'j) \to \beta \oplus ('k \ominus 1)\,n \oplus 'j \\
\textit{Я}.
\end{gathered}$$

Эта программа представляет собой тройной циклический процесс.

Пример 2. *Вычисление суммы*

$$S = \sum_{i=1}^{n} \sum_{j=i}^{m} a_{ij}.$$

Предполагается, что величины a_{ij} ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$) — элементы некоторой прямоугольной матрицы, размещенные строками по адресам $\alpha \oplus 1$, $\alpha \oplus 2$, $\ldots$, $\alpha \oplus mn$.

Задача состоит в просмотре адресов, которые содержат элементы прямоугольной матрицы, размещенные по диагонали и над нею. При этом содержимое каждого такого адреса прибавляется к сумме предыдущих; исходное значение суммы предыдущих равняется нулю. Допустим, что суммирование производится по строкам, начиная с первой.

Пусть адрес φ содержит адреса диагональных членов матрицы; вначале $'\varphi = \alpha \oplus 1$, т. е. φ отмечает первый диагональный элемент; адрес ψ содержит «текущий элемент» строки, начиная от диагонального; адрес $\psi 1$ содержит адреса последнего столбца матрицы; вначале $'\psi 1 = \alpha \oplus m$, т. е. $\psi 1$ содержит адрес последнего элемента первой строки матрицы, а также $'\varphi 1 = \alpha \oplus mn$; в γ содержится результат вычислений (вначале $'\gamma = 0$). Тогда получим

$$\begin{gathered}
B\ \ldots\ '\varphi \to \psi \\
'\gamma + '\psi \to \gamma \\
'\psi \oplus 1 \to \psi \\
\textbf{Р}\,\{'\psi \leqslant '\psi 1\}\,B \\
'\varphi \oplus m \oplus 1 \to \varphi;\ '\psi \oplus m \to \psi 1 \\
\textbf{Р}\,\{'\varphi \leqslant '\varphi 1\}\,B \\
\textit{Я}.
\end{gathered}$$

$$Ц\,(1\ (1)\ n \to i)$$
$$Ц\,('i\ (1)\ m \to j)$$
$$'\,(\alpha \oplus ('i \ominus 1)\ m \oplus 'j) + '\gamma \to \gamma$$
$$И.$$

Заметим, что в данную схему легко вкладывается каждая задача, решение которой сводится к просмотру тех же ячеек.

Пример 3. *Вычисление минора матрицы.* Пусть $A = \{a_{ij}\}$ квадратная матрица, заданная по строкам в виде α-последовательности так, что $'\alpha = n$ — порядку матрицы. Требуется в β-последовательность поместить минор элемента a_{kl}, а по адресу β — его порядок $n - 1$. Числа k и l заданы как $'K$ и $'L$.

Примем следующий порядок работы. Элементы матрицы просматриваются по строкам; если номер строки равен $'K$, то строка пропускается; если строка не выбрасывается, то ее элементы последовательно просматриваются и не принадлежащие $'L$-му столбцу переписываются в β-последовательность. В результате получим программу

$$0 \to s$$
$$Ц\,(1\ (1)\ '\alpha \to i)\ M$$
$$Р\,('i = 'K)\ M1$$
$$Ц\,(1\ (1)\ '\alpha \to j)\ M1$$
$$Р\,('j = 'L)\ M2$$
$$'s \oplus 1 \to s$$
$$'(\alpha \oplus ('i \ominus 1) \otimes '\alpha \oplus 'j) \to \beta \oplus 's$$
$$M2\ldots\ M1\ldots$$
$$M\ldots$$
$$'\alpha \ominus 1 \to \beta;\ И$$

Пример 4. *Решение уравнения теплопроводности*

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} \tag{3.38}$$

в области $0 \leqslant t \leqslant T;\ 0 \leqslant x \leqslant X$ *с начальными и граничными условиями*

$$u(0,\ x) = f(x);\ u(t,\ 0) = \varphi(t);\ u(t,\ X) = \psi(t)$$

методом сеток с запоминанием получаемых значений функции $u(t, x)$.

Положим $\Delta x = h;\ \Delta t = \dfrac{h^2}{2a^2}$ и обозначим

$$K = \frac{2a^2}{h^2}\,T;\ \ I = \frac{X}{h};\ u_{kl} = u(k\Delta t,\ ih)$$
$$(k = 0,\ 1,\ \ldots,\ K;\ i = 0,\ 1,\ \ldots,\ I).$$

Заменив

$$\frac{\partial u}{\partial t} \cong \frac{1}{\Delta t}\,(u_{k+1,\,i} - u_{k,\,i});\quad \frac{\partial^2 u}{\partial x^2} \cong \frac{1}{h^2}\,(u_{k,\,i+1} - 2u_{k,\,i} + u_{k,\,i-1}),$$

решение уравнения (3.38) сведем к решению разностного уравнения

$$u_{k+1,\,i} = \frac{1}{2}(u_{k,\,i-1} + u_{k,\,i+1}),\ k = 0,\ 1,\ \ldots,\ K-1;$$
$$i = 1,\ 2,\ \ldots,\ I-1. \qquad (3.39)$$

Для значений $u_{k,\,i}$ $(k = 0,\ 1,\ \ldots,\ K;\ i = 0,\ 1,\ \ldots,\ I)$ выделим последовательность адресов

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha \oplus (I \oplus 1)(K \oplus 1), \qquad (3.40)$$

причем

$$u_{kl} = {}'(\alpha \oplus i \oplus 1 \oplus k(I \oplus 1)).$$

Предположим, что значения функции $u(t,\ x)$ в граничных узлах сетки заранее подсчитаны и внесены в соответствующие адреса, т. е

$$'(\alpha \oplus i \oplus 1) = u_{0i} = f(ih),\ (i = 0,\ 1,\ \ldots,\ I)$$
$$'(\alpha \oplus 1 \oplus k(I \oplus 1)) = u_{k0} = \varphi(k\Delta t),\ (k = 1,\ 2,\ \ldots,\ K) \qquad (3.41)$$
$$'(\alpha \oplus (1 \oplus k)(I \oplus 1)) = u_{kI} = \psi(k\Delta t),\ (k = 1,\ 2,\ \ldots,\ K).$$

Составим адресный алгоритм, обеспечивающий вычисление искомых значений u_{kl} и их размещение (запоминание) по остальным адресам последовательности (3.40).

Для величин $u_{k,\,i-1}$ введем адрес второго ранга ω. Вначале при $k = 0$, $i = 1$

$$'\omega = \alpha \oplus 1,\ \text{т. е. } {}^2\omega = u_{0,\,0}.$$

Согласно формуле (3.39) по адресу ω можно определить адреса второго ранга двух других значений функции u, входящих в эту формулу:

$$u_{k,\,i+1} = '('\omega \oplus 2).$$
$$u_{k+1,\,i} = '('\omega \oplus 1 \oplus I \oplus 1) = '('\omega \oplus I \oplus 2).$$

Примем следующий порядок вычислений. По значениям функции u_{0i} на первой вертикали вычислим значения u_{1i} на следующей вертикали, начиная с точки, соответствующей значению $i = 1$, до точки $i = I - 1$; затем переходим к вычислениям при $k = 2$ и т. д.

Для определения окончания вычислений по вертикали введем адрес второго ранга $\omega 1$ для одного из значений u_{ik}, используемых в формуле (3.39), например для $u_{k,\,i-1}$. Таким образом, вначале

$$'\omega 1 = \alpha \oplus I \ominus 1,\ {}^2\omega 1 = u_{0,\,I-2}.$$

При переходе на следующую вертикаль содержимое $\omega 1$ должно быть увеличено на $I \oplus 1$.

Для определения окончания вычислений введем адрес второго ранга $\omega 2$ для последнего используемого в формуле (3.39) значения $u_{k,\,i-1}$

$$'\omega 2 = \alpha \oplus (I \oplus 1)K \ominus 2,\ {}^2\omega 2 = u_{K-1,\,I-2}.$$

Для принятого исходного адресного отображения получим адресную программу

$$B \ldots \frac{1}{2}(^2\omega \oplus {}'('\omega \oplus 2)) \rightarrow {}'\omega \oplus I \oplus 2;$$
$${}'\omega \oplus 1 \rightarrow \omega$$
$$P\{{}'\omega \leqslant {}'\omega 1\}\, B$$
$${}'\omega 1 \oplus I \oplus 1 \rightarrow \omega 1$$
$$P\{{}'\omega 1 \leqslant {}'\omega 2\}\, B \downarrow !,$$

или

$$Ц\{\omega 1\,(I \oplus 1)\,\omega 2 \rightarrow \omega 1\},!$$
$$Ц\{{}'\omega\,(1)\,{}'\omega 1 \rightarrow \omega\}$$
$$\frac{1}{2}(^2\omega \oplus {}'('\omega \oplus 2)) \rightarrow {}'\omega \oplus I \oplus 2.$$

Пример 5. *Определение принадлежности блуждающей точки границе области.*

Для простоты рассмотрим двумерную область G с границей Γ. Пусть блуждающая точка находится внутри области и необходимо определить ее выход на границу. Предлагаемый ниже алгоритм может быть перенесен на области большего числа измерений. Предположим, что граница области задана таблично. В случае аналитического задания ее следует предварительно оценить время, необходимое для проверки принадлежности точки внутренней или внешней области Γ, например, по знаку функции, определяющей уравнение границы в данной точке; первое попадание во внешнюю область можно принять за достижение границы. Если такая проверка ввиду сложности аналитического задания границы Γ сопряжена с большими затратами времени, то целесообразно свести задачу к случаю табличного задания границы области[1].

Не ограничивая общности, можно предположить, что числа x, y — целые и граница определяется с точностью до шага, равного единице по каждой оси. Тогда множества возможных значений x и y, принадлежащих границе, будут некоторые целые числа

$$x = x_0,\ x_0 + 1,\ \ldots,\ x_0 + N_1 - 1; \tag{3.42}$$
$$y = y_0,\ y_0 + 1,\ \ldots,\ y_0 + N_2. \tag{3.43}$$

Для каждого значения x_i $(i = 1, 2, \ldots, N_1)$ выберем те y_{ij} $(1 \leqslant j \leqslant n_i)$, которые в паре с x_i определяют граничные точки Γ. В каждой из полученных N_1 групп упорядочим значения y, например, в порядке убывания $y_{ij} > y_{ij+1}$. Упорядоченный набор координат разместим по адресам

$${}'(\omega_i \oplus j) = y_{ij};\ 1 \leqslant i \leqslant N_1;\ 1 \leqslant j \leqslant n_i.$$

Введем адреса для адресов $\omega_i \oplus 1$ $(i = 1, 2, \ldots, N_1)$:

$${}'(s \oplus x_0 \oplus k) = \omega_k \oplus 1 \quad (1 \leqslant k \leqslant N_1).$$

---

[1] Для случая неявного задания функции, определяющей границу области, в работе [2] приведена экономная методика перевода аналитического задания функции в табличное.

Введем адреса γ, β для координат блуждающей точки

$$'\alpha = x\,(=x_0 + k); \quad \beta = y\,(=y_0 + r).$$

Проверка принадлежности точки (x, y) границе области может быть реализована следующей адресной программой, в которой входная строка помечена меткой B. При попадании на границу осуществляется выход по метке B_1, в противном случае — по метке B_2:

$$\begin{aligned} A \;&\ldots\; '('\alpha \oplus s) \oplus 1 \to '\alpha \oplus s \\ B \;&\ldots\; \boldsymbol{P}\,\{^2('\alpha \oplus s) > '\beta\}\; A \\ &\quad\; \boldsymbol{P}\,\{^2('\alpha \oplus s) = '\beta\}\; B_1 \downarrow B_2. \end{aligned}$$

Поскольку

$$'\alpha = x,\ '('\alpha \oplus s) = \omega_k \oplus 1,\ '(\omega_k \oplus 1) = {}^2('\alpha + s) = y_{k1},$$

то по входной строке проверяется выполнение неравенства $y_{k1} > y$. При выполнении этого неравенства совершается переход к строке с меткой A, а в противном случае — к третьей строке. Первая строка означает переход в таблице к следующему y. Последняя строка проверяет условие

$$y_{kj} = y;$$

при выполнении этого условия алгоритм заканчивает работу выходом на метку B_1 — точка принадлежит границе области. В противном случае работа заканчивается выходом на метку B_2, так как точка (x, y) оказалась внутренней точкой области; блуждания должны быть продолжены.

При многократном выполнении блужданий работа, затраченная на «предварительное изучение» структуры границы (т. е. на выделение подмножеств y_{ij} для каждого фиксированного x_i, $i=1, 2, \ldots, N_1$), и затрата памяти на хранение информации о ней (адреса $s \oplus x_0 + k$, $k = 1, 2, \ldots, N_1$) вполне себя оправдывают. Алгоритм, реализующий «предварительное изучение» структуры границы (т. е. выделение и упорядочение подмножеств y_{ij}) с соответствующим формированием информации об этой структуре (в адресах $x \oplus x_0 \oplus k$), также может быть легко реализован на машине.

Изложенный алгоритм легко обобщить на ряд аналогичных задач, таких как задача поиска слова в словаре, поиска в таблице значения функции одного или многих переменных и др. При этом нетрудно учесть размещение таблиц или словаря во внешней памяти ЗУ.

Пример 6. *Алгоритм распознавания некоторых свойств последовательностей символов* [4]. Введем некоторые определения. Пусть $\alpha = \{a_1, \ldots, a_n\}$ — алфавит исходных символов, $S(\alpha)$ — множество всех слов в алфавите α.

Совокупность символов

$$T = \begin{matrix} t_{11} & t_{21} & \ldots & t_{p1} \\ t_{12} & t_{22} & \ldots & t_{p2} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ t_{1h_1} & t_{2h_2} & \ldots & t_{ph_p}, \end{matrix}$$

где $t_{jk} \in \alpha$ для $1 \leqslant j \leqslant p$, $1 \leqslant k \leqslant h_j$ называется таблицей в алфавите α, число $p \geqslant 2$ — длиной таблицы. Множества

$$t_j = \bigcup_{k=1}^{h_j} \{t_{jk}\} \quad (1 \leqslant j \leqslant p)$$

называются столбцами таблицы.

Множество всех слов длины p

$$X = x_1 \ldots x_p, \ (x_j \in \alpha),$$

для которых $x_j \in t_j$ называется множеством слов, соответствующих таблице T, и обозначается через $\tilde{T}$. Считаем, что слово $Y \in S(\alpha)$ имеет простое свойство T, если некоторое слово $X \in T$ входит в слово Y, т. е. $Y = EXF$, где $X \in \tilde{T}$, $E \in S(\alpha)$, $F \in S(\alpha)$.

Изображением таблицы T называется булева матрица B (размерности $n \times p$), элементы которой определяются соотношением

$$b(i, j) = \begin{cases} 1, \text{ если } a_i \in t_j \ (1 \leqslant i \leqslant n; \quad 1 \leqslant j \leqslant p) \\ 0, \text{ в противном случае.} \end{cases}$$

Строки матрицы B обозначаются через $B(1), \ldots, B(n)$ и рассматриваются как булевы векторы размерности p. Логические (покомпонентные) и арифметические операции над подобными векторами определяются как (поразрядные) операции над словами некоторой двоичной вычислительной машины. Пусть $E_1 = (1, 0, \ldots, 0)$ (размерности p).

Слово $Y = y_1 \ldots y_q = a_{s_1} \ldots a_{s_q}$ ($y_r = a_{s_r} \in \alpha$, $1 \leqslant r \leqslant q$) имеет простое свойство T только тогда, когда среди векторов (размерности p)

$$Q_0 = 0 \ldots 0$$

$$Q_{r+1} = \left(Q_r \cdot \frac{1}{2} \vee E_1\right) \wedge B(s_{r+1})$$

найдется такой вектор Q_r ($1 \leqslant r \leqslant q$), последняя ($p$-я) координата которого равна 1, т. е. $Q_r \wedge E_p \neq 0$, где $E_p = (0, 0, \ldots, 0, 1)$.

Для простоты считаем, что буквы алфавита α закодированы натуральными числами 1, 2, ..., n. Пусть слово X задано в виде α-последовательности ($'\alpha = q$ — длина слова)

$$'(\alpha \oplus i) = a_{s_i} = y_i.$$

Пусть строки булевой матрицы B закодированы в виде β-последовательности: $'\beta = n$, $'(\beta \oplus i) = B(i)$, причем l-й компоненте вектора $B(i)$ соответствует l-й разряд ячейки $(\beta \oplus i)$. Обозначение: $'(\beta \oplus i)_l = B_l(i)$.

Выделим для хранения булевых векторов Q_i адрес Q (умножение булевого вектора на $\frac{1}{2}$ означает сдвиг всех разрядов влево на один разряд и засылку нуля в крайний правый разряд). Пусть по адресу k содержится вектор (1, 0, ..., 0), а по адресу v — вектор (0, ..., 0, 1). Пусть фиксатор φ обозревает буквы данного слова Y. Вначале $'\varphi = \alpha$.

Теперь программа распознавания простого свойства T может быть записана в следующем виде:

$$Ц\,\{1\,(1)'\alpha \to 1\}\,M$$
$$('Q \times 2^{-1}\,\mathrm{V}\,'k) \wedge\,'(\beta \oplus\,'(\alpha \oplus' 1))\to Q$$
$$P\,\{'Q \wedge' v = 0\} \downarrow 1Я$$
$$M\ \ldots\ 2Я$$

Выход $1Я$ означает наличие простого свойства в данном слове, выход $2Я$ — его отсутствие.

Введенные понятия обобщаются на случай нескольких таблиц. Пусть $T = \{T_1, \ldots, T_m\}$ — система таблиц в алфавите α; длиной системы называется число $p = p_1 + \ldots + p_m$. Изображением системы называется булева матрица $n \times p$-го порядка $B = B_1B_2 \ldots B_m$, получаемая в результате последовательного выписывания матриц B_i. Векторами u и v называются булевы векторы, имеющие в качестве своих компонент с номерами соответственно

$$1,\ p_1 + 1,\ p_1 + p_2 + 1,\ \ldots,\ p_1 + p_2 + \ldots + p_{m-1} + 1 \qquad (u)$$

и

$$p_1,\ p_1 + p_2,\ \ldots\ p_1 + p_2 + \ldots + p_m \qquad (v)$$

единицы, а остальные компоненты, равные нулю.

Можно показать, что слово Y тогда и только тогда обладает простым свойством T_i $(1 \leqslant i \leqslant m)$, если среди векторов

$$Q_0 = 0$$

$$Q_{r+1} = \left(Q_r \times \frac{1}{2} \vee u\right) \wedge B\,(S_{r+1})$$

найдется такой вектор Q_r $(1 \leqslant r \leqslant q)$, имеющий своей $p_1 + \ldots + p_i$ координатой единицу. Если для всех Q_r $(1 \leqslant r \leqslant q)$

$$Q_r \wedge v = 0,$$

слово не имеет ни одного из простых свойств системы.

8. ОБ АЛГОРИТМИЗАЦИИ ПЕРЕХОДА С ОБЩЕАЛГОРИТМИЧЕСКОГО УРОВНЯ НА УРОВЕНЬ УСЛОВНЫХ АДРЕСОВ

Рассмотрим задачу перевода алгоритмов с общеалгоритмического уровня на уровень условных адресов, принятый в качестве основного.

Для этого необходимо алгоритмически описать операции следования в множестве адресов. На общеалгоритмическом уровне эти операции обычно описываются с помощью индексов. Поэтому алгоритмизация операций следования сводится к нахождению алгоритмов определения адресов по заданным индексам элементов.

Алгоритмы определения адресов элементов по их индексам зависят от способа кодирования информации;

стандартизация этих алгоритмов для различных наиболее употребительных способов кодирования существенно облегчит задачу программирования.

Поясним сказанное примером.

Усовершенствованный метод Гаусса для симметричной матрицы [18].

Пусть $A = \{a_{ij}\}$; $i, j = 1, 2, \ldots, n$, $a_{ij} = a_{ji}$ — исходная матрица и $F\{a_{i,\,n+1}\}$ — вектор правых частей.

Алгоритм состоит из двух этапов — прямой ход и обратный ход.

Прямой ход. Элементы a_{ij} преобразуем последовательно по строкам согласно рекурентным формулам

$$\bar{a}_{1j} = a_{1j},\ 1 \leqslant j \leqslant n+1$$

$$\bar{a}_{ij} = a_{ij} - \sum_{k=1}^{i-1} \frac{\bar{a}_{ki} \cdot \bar{a}_{kj}}{\bar{a}_{kk}},\ 2 \leqslant i \leqslant n,\ i \leqslant j \leqslant n+1.$$

В результате получим треугольную матрицу.

Обратный ход — решение системы уравнений с треугольной матрицей $\{a_{ij}\}$; $i = 1, 2, \ldots, n$, $i \leqslant j \leqslant n$.

Расчетные формулы имеют вид

$$x_i = \frac{\bar{a}_{i,\,n+1} - \sum_{k=i+1}^{n} \bar{a}_{ik} x_k}{\bar{a}_{ii}} \quad (i = n,\ n-1,\ \ldots,\ 1).$$

1. Составление программ на общеалгоритмическом уровне. Вновь получаемым элементам $\bar{a}_{ij}$ могут быть поставлены в соответствие те же адреса, в которых хранились элементы a_{ij}.

Приняв в качестве исходного отображения

$$'\alpha_{ij} = a_{ij}\ (i = 1, 2, \ldots, n;\ 1 \leqslant j \leqslant n+1)$$

s_j — адреса, отведенные для компонент вектора решений $x_i (i = 1, 2, \ldots, n)$, алгоритмы прямого и обратного ходов можно представить в виде

$$\begin{array}{l} \textit{Прямой ход...} \\ \textbf{Ц}\{2\,(1)\,n \to i\},\ \textbf{Я} \\ \textbf{Ц}\{i\,(1)\,n+1 \to j\} \\ \textbf{Ц}\{1\,(1)\,i-1 \to k\} \\ '\alpha_{ij} - \dfrac{'\alpha_{ki} \cdot '\alpha_{kj}}{'\alpha_{kk}} \Rightarrow \alpha_{ij} \end{array} \qquad (3.44)$$

Результативное множество адресов совпадает с исходным.

Обратный ход...

$$\begin{array}{l} \boldsymbol{Ц}\{n(\ominus 1)\,1 \to i\}\,\boldsymbol{M},\ \boldsymbol{Я} \\ {}'\alpha_{i,\,n+1} \Rightarrow s_i \\ \boldsymbol{Ц}\{'i \oplus 1\,(1)\,n \to k\} \\ {}'s_i - {}'\alpha_{ik} \cdot {}'s_k \Rightarrow s_i \\ \dfrac{{}'s_i}{{}'\alpha_{ii}} \Rightarrow s_j \\ \boldsymbol{M}\ \ldots \end{array} \tag{3.45}$$

Результативное множество адресов s_i $(i = 1, 2, \ldots, n)$.

2. Уровень условных адресов. Для перехода от программ (3.44) и (3.45), составленных на общеалгоритмическом уровне, к программам на уровне условных адресов необходимо при избранном способе кодирования информации — исходном адресном отображении — по индексам элементов найти их адреса.

Рассмотрим несколько способов кодирования.

А. Матрица задана последовательностью адресов, соответствующих ее элементам при обозревании по строкам или по столбцам: $\alpha \oplus 1$, $\alpha \oplus 2$, ..., $\alpha \oplus n \otimes (n \oplus 1)$.

В первом случае

$$a_{ij} = {}'(\alpha \oplus (i \ominus 1) \otimes (n \oplus 1) \oplus j). \tag{3.46}$$

Во втором случае

$$a_{ij} = {}'(\alpha \oplus (j \ominus 1) \otimes n \oplus i). \tag{3.47}$$

В связи с этим для соотношения (3.46) программа (3.44) приобретает вид:

$$\begin{array}{l} \textit{Прямой ход 1а}\ \ldots\ \boldsymbol{Ц}\{2\,(1)\,n \Rightarrow I\},\ \boldsymbol{Я} \\ \qquad\qquad \boldsymbol{Ц}\{'I\,(1)\,n \oplus 1 \Rightarrow J\} \\ \qquad\qquad \boldsymbol{Ц}\{1\,(1)'\,I \ominus 1 \Rightarrow K\} \\ \qquad\qquad {}'(\alpha \oplus ('I \ominus 1) \otimes (n \oplus 1) \oplus J) - \\ - \dfrac{{}'(\alpha \oplus ({}'K \ominus 1) \otimes (n \oplus 1) \oplus {}'I)' \cdot (\alpha \oplus ({}'K \ominus 1) \otimes (n \oplus 1) \oplus {}'J)}{{}'(\alpha \oplus ({}'K \ominus 1) \otimes (n \oplus 1) \oplus {}'K)} \Rightarrow \\ \qquad\qquad \Rightarrow \alpha \oplus ('I - 1) \otimes (n \oplus 1) \oplus {}'J \end{array} \tag{3.48}$$

и для соотношения (3.47)

$$\begin{gathered}\textit{Прямой ход 1б} \ldots \boldsymbol{Ц}\{2\,(1)\,n \Rightarrow I\},\ \textit{Я} \\ \boldsymbol{Ц}\{'I\,(1)\,n \oplus 1 \Rightarrow J\} \\ \boldsymbol{Ц}\{1\,(1)'\,I \oplus 1 \Rightarrow K\} \\ '(\alpha \oplus ('J \ominus 1) \otimes n \oplus 'I) - \\ - \frac{'(\alpha \oplus ('I \ominus 1) \otimes n \oplus 'K)\,'(\alpha \oplus ('J \ominus 1) \otimes n \oplus 'K)}{'(\alpha \oplus ('K \ominus 1)\,n \oplus 'K)} \Rightarrow \\ \Rightarrow \alpha \oplus ('J \ominus 1)\,n \oplus 'I.\end{gathered} \tag{3.49}$$

Аналогично можно переписать программу обратного хода.

Б. Матрица задана диагональными и наддиагональными элементами по столбцам, при этом, например, для $n = 4$ размещение исходной информации по адресам имеет вид

$'(\alpha \oplus 1) = a_{11}$	$'(\alpha \oplus 2) = a_{12}$	$'(\alpha \oplus 4) = a_{13}$	$'(\alpha \oplus 7) = a_{14}$	$'(\alpha \oplus 11) = a_{15}$
	$'(\alpha \oplus 3) = a_{22}$	$'(\alpha \oplus 5) = a_{23}$	$'(\alpha \oplus 8) = a_{24}$	$'(\alpha \oplus 12) = a_{25}$
		$'(\alpha \oplus 6) = a_{33}$	$'(\alpha \oplus 9) = a_{34}$	$'(\alpha \oplus 13) = a_{35}$
			$'(\alpha \oplus 10) = a_{44}$	$'(\alpha \oplus 14) = a_{45}$

(3.50)

Порядковый номер элемента информации a_{ij} в последовательности

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha \oplus \frac{n^2 \oplus n}{2} \oplus n$$

будем называть его приведенным индексом $\pi(i, j)$.

Нетрудно убедиться, что

$$\pi(i, j) = i \oplus \sum_{s=1}^{j-1} s \tag{3.51}$$

или

$$\pi(i, j) = i \oplus \frac{j}{2} \otimes (j \ominus 1). \tag{3.52}$$

Для записи алгоритмов на уровне условных адресов можно использовать каждую из формул (3.51) и (3.52). В случае применения формулы (3.51) получаем программу прямого хода:

$$
\begin{gathered}
\textit{Прямой ход 2 } \ldots \ \boldsymbol{Ц}\{2\,(1)\,n \Rightarrow i\} \\
\boldsymbol{Ц}\{'i\,(1)\,n \oplus 1 \Rightarrow j\} \\
\boldsymbol{Ц}\{'i\,(1)\,n \oplus 1 \Rightarrow i\} \\
\boldsymbol{Ц}\{1\,(1)'\,i \ominus 1 \Rightarrow k\}\,M \\
'i \Rightarrow r_1 \\
\boldsymbol{Ц}\{1\,(1)'\,j \ominus 1 \Rightarrow s\} \\
'r_1 \oplus 's \Rightarrow r_1 \\
'k \Rightarrow r_2 \\
\boldsymbol{Ц}\{1\,(1)'\,i \ominus 1 \Rightarrow s\} \\
'r_2 \oplus 's \Rightarrow r_2 \\
'k \Rightarrow r_3 \\
\boldsymbol{Ц}\{1\,(1)'j \ominus 1 \Rightarrow s\} \\
'r_3 \oplus 's \Rightarrow s \\
'k \Rightarrow r_4 \\
\boldsymbol{Ц}\{1\,(1)'k \ominus 1 \Rightarrow s\} \\
'r_4 \oplus 's \Rightarrow r_4 \\
'(\alpha \oplus 'r_1) - \frac{'(\alpha \oplus 'r_2) \cdot '(\alpha \oplus 'r_3)}{'(\alpha \oplus 'r_4)} \Rightarrow \alpha \oplus 'r_1 \\
M \ \ldots
\end{gathered}
\tag{3.53}
$$

При использовании формулы (3.52) та же программа переписывается в виде:

$$
\begin{gathered}
\textit{Прямой ход 2 } \ldots \ \boldsymbol{Ц}\{2\,(1)''\varphi \Rightarrow i\},\ \boldsymbol{Я} \\
\boldsymbol{Ц}\{'i\,(1)''\varphi \ \oplus 1 \Rightarrow j\} \\
\boldsymbol{Ц}\{1\,(1)'i \ominus 1 \Rightarrow k\} \\
'('\varphi \oplus 'i \oplus \tfrac{'j}{2} \otimes ('j \ominus 1)) - \\
- \frac{'\left('\varphi \oplus 'k \oplus \tfrac{'i}{2} \otimes ('i \ominus 1)\right) \cdot '\left('\varphi \oplus 'k \oplus \tfrac{'j}{2} \otimes ('j \ominus 1)\right)}{'\left('\varphi \oplus \tfrac{'k}{2}\,('k \oplus 1)\right)} \Rightarrow \\
\Rightarrow '\varphi \oplus 'i \oplus \tfrac{'j}{2} \otimes ('j \ominus 1).
\end{gathered}
\tag{3.54}
$$

Несмотря на простоту записи программы (3.54) по сравнению с записью (3.53), в ней требуется для каждого значения $'j$ вычислять выражение

$$'i \oplus \frac{'j \otimes (j \ominus 1)}{2},$$

где $'i$ и $'j$ — целые числа (индексы); операции над индексами, в зависимости от особенности машин, могут выполняться более или менее сложно. Поэтому в отдельных случаях запись (3.53) может оказаться более выгодной.

Для возможности получения результата — вектора решений — в программе обратного хода в произвольном массиве адресов введем адрес ψ:

$$'\psi = l$$

и потребуем, чтобы размерность вектора решений n помещалась в процессе реализации алгоритма по адресу l.

Используя зависимость (3.52), получаем программу обратного хода:

$$\begin{gathered}
\textit{Обратный ход} \ \ldots \ ''\varphi \Rightarrow '\psi \\
\textit{Ц}\{''\varphi\,(\ominus 1)\,1 \Rightarrow i\}\,M,\ \textit{Я} \\
'\left('\varphi \oplus 'i \oplus \frac{''\varphi \oplus 1}{2} \otimes ''\varphi\right) \Rightarrow '\psi \oplus 'i \\
\textit{Ц}\{'i \oplus 1\,(1)''\varphi \Rightarrow k\} \\
'('\psi \ominus 'i) - '\left('\varphi \oplus 'i \oplus \frac{'k}{2} \otimes (k \ominus 1)\right)'('\psi \oplus 'k) \Rightarrow '\psi \oplus 'i \qquad (3.55) \\
\frac{'('\psi \oplus 'i)}{\left('\varphi \oplus \dfrac{'i \otimes ('i \oplus 1)}{2}\right.} \Rightarrow '\psi \oplus 'i \\
M \ \ldots
\end{gathered}$$

Множество результативных адресов:

$$l \oplus i\ (i = 1,\ 2,\ \ldots,\ n).$$

В. Матрица задана диагональными и наддиагональными элементами по строкам

$$\pi(i,\ j) = j \oplus \sum_{k=1}^{i-1} (n \ominus k \oplus 1), \qquad (3.56)$$

или

$$\pi(i, j) = j \oplus \left(n \oplus 1 \ominus \frac{i}{2}\right) \otimes (i \ominus 1). \qquad (3.57)$$

Используя соотношение (3.56), приведем для этого случая программу прямого хода, которая понадобится далее.

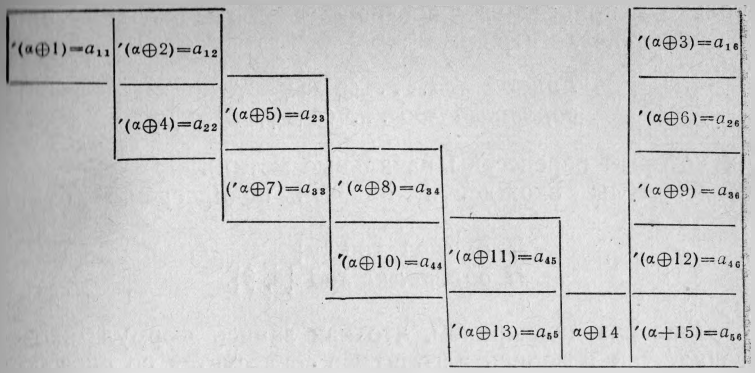
$$\begin{array}{c}
\textit{Прямой ход 3} \ldots \boldsymbol{Ц}\{2(1)\, n \Rightarrow i\}, Я \\
\boldsymbol{Ц}\{'i\,(1)\, n \oplus 1 \Rightarrow j\} \\
\boldsymbol{Ц}\{1\,(1)'i \ominus 1 \Rightarrow k\}\, M \\
'j \Rightarrow r_1 \\
\boldsymbol{Ц}\{1\,(1)'\, i \ominus 1 \Rightarrow s\} \\
'r_1 \oplus (n \ominus 's \oplus 1) \Rightarrow r_1 \\
'i \Rightarrow r_2 \\
\boldsymbol{Ц}\{1\,(1)'k \ominus 1 \Rightarrow s\} \\
'r_2 \oplus (n \ominus 's \oplus 1) \Rightarrow r_2 \\
'j \Rightarrow r_3 \\
\boldsymbol{Ц}\{1\,(1)'k \ominus 1 \Rightarrow s\} \\
'r_3 \oplus (n \ominus 's \oplus 1) \Rightarrow r_3 \\
'k \Rightarrow r_4 \\
\boldsymbol{Ц}\{1\,(1)'k \ominus 1 \Rightarrow s\} \\
'r_4 \oplus (n \ominus 's \oplus 1) \Rightarrow r_4 \\
MS \ldots {}'(\alpha \oplus 'r_1) - \frac{'(\alpha \oplus 'r_2) \cdot '(\alpha \oplus 'r_3)}{'(\alpha \oplus 'r_4)} \Rightarrow \alpha \oplus 'r_1 \\
M \ldots
\end{array} \qquad (3.58)$$

Г. Матрица симметрична и ее элементы, отличные от нуля, расположены в виде полосы вдоль диагонали. Заданы (в целях «экономии» памяти) лишь диагональные и наддиагональные элементы последовательностью адресов по строкам.

Пусть

$$\pi(i, j) = \begin{cases} (m \oplus 1) \otimes i, & \text{если } j = n \oplus 1: \\ j \oplus (i \ominus 1) \otimes m, & \text{если } i \leqslant j \leqslant n. \end{cases}$$

Здесь n — порядок матрицы, m — номер крайней диагонали (параллельной главной), элементы которой отличны от нуля. Так, для $n = 5$, $m = 2$ адресное соответствие имеет вид показанной таблицы.



(Адрес $\alpha \oplus 14$ в алгоритме не используется).

Составим программу прямого хода в этом случае.

Согласно программе (3.44) нам понадобятся значения приведенного индекса $\pi(i, j)$ для следующих пар аргументов:

$$(i, j),\ (k, i),\ (k, j),\ (k, k).$$

Так как для индексов i, j, k согласно формулам циклирования в программе (3.44) выполняются соотношения

$$i \leqslant n;\ k \leqslant i - 1 < n,$$

то

$$\pi(k, i) = i \oplus (k \ominus 1) \otimes m;$$
$$\pi(k, k) = k \oplus (k \ominus 1) \otimes m.$$

Таким образом, программу прямого хода в данном случае можно записать в виде

$$\textbf{Ц}\{2\,(1)\,n \to i\}$$
$$\textbf{Ц}\{'i\,(1)\,n \oplus 1 \to j\}$$
$$\textbf{Ц}\{1\,(1)\,i \ominus 1 \to k\}\,L$$
$$\textbf{Р}\{j = n \oplus 1\}\,(m \oplus 1) \otimes i \Rightarrow r_1,\ (m \oplus 1) \otimes k \Rightarrow r_2 \downarrow j \oplus$$
$$\oplus\,(i \ominus 1) \otimes m \Rightarrow r_1,\ j \oplus (k \ominus 1) \otimes k \Rightarrow r_2$$
$$'(\alpha \oplus 'r_1) - \frac{'(\alpha \oplus i \oplus (k \ominus 1) \otimes m) \cdot '(\alpha \oplus 'r_2)}{'(\alpha + k \oplus (k \ominus 1) \otimes m)} \Rightarrow \alpha + 'r$$
$$L\ \ldots\ Я.$$

Каждый из приведенных алгоритмов можно оформить в виде подпрограммы. При принятом способе кодирования исходной матрицы и ее порядка для этого остается

только к приведенным программам, соответственно, прибавить вначале строки

$$\textit{прямой ход} \ldots \varnothing \Rightarrow \varphi$$
$$\textit{обратный ход} \ldots \varnothing \Rightarrow \varphi,\ \varnothing \Rightarrow \psi$$

(к которым перенесены начальные метки).

Формулы вхождения на эти подпрограммы будут иметь вид

$$\boldsymbol{П}\ \textit{прямой ход}\ \{\alpha\}$$
$$\boldsymbol{П}\ \textit{обратный ход}\ \{\alpha, \beta\}.$$

Еще раз подчеркнем, что как запись формул вхождения, так и запись алгоритмов согласуется со способом кодирования информации.

9. О ПРЕОБРАЗОВАНИИ АДРЕСНЫХ ПРОГРАММ

При записи алгоритмов в адресном виде над ними можно производить эквивалентные преобразования.

Под эквивалентными алгоритмами понимаем такие два алгоритма, которые для одного и того же исходного адресного отображения в результате своей реализации дают одинаковые отображения на результирующее множество адресов (различие отображений в остальной части множества адресов роли не играет). Преобразования алгоритмов, сохраняющие их эквивалентность, называем эквивалентными.

Следует ожидать, что в результате развития работ по теории алгоритмов будут найдены наборы эквивалентных преобразований и сформулированы правила для направленных преобразований алгоритмов в целях минимизации числа символов, необходимых для записи алгоритма в языке, минимизации числа элементарных шагов работы алгоритма (числа тактов ИАВМ), исключения из записи алгоритма тех или иных символов в связи с необходимостью удовлетворить требования стиля входного языка некоторой программирующей программы и др. Кроме того, можно ставить вопрос об оптимизации адресных алгоритмов в данном стиле с тем, чтобы перевод их с помощью той или иной программирующей программы обеспечивал бы получение оптимальных рабочих программ.

Рассмотрим два примера.

Пример 1. Используем приведенную в предыдущем параграфе программу (3.58) прямого хода (случай кодирования треугольной матрицы по строкам). Осуществим эквивалентные преобразования этой программы, сокращающие число тактов работы алгоритма.

1. Так как параметры циклов s (с помощью которых вычисляются приведенные индексы элементов a_{ki}, a_{kj}, a_{kk}) изменяются в одинаковых пределах и получаемые величины $'r_2$, $'r_3$, $'r_4$ обозреваются одновременно (строка с меткой *MS*), то образование этих величин можно осуществить одним циклом

$$'i = r_2;\ 'j \Rightarrow r_3;\ 'k \Rightarrow r_4$$
$$\boldsymbol{Ц}\{1\,(1)\,k \ominus 1 \Rightarrow s\}$$
$$'r_2 \oplus (n \ominus 's \oplus 1) \Rightarrow r_2;\ 'r_3 \oplus (n \ominus 's \oplus 1) \Rightarrow r_3;$$
$$'r_4 \oplus (n - 's \oplus 1) \Rightarrow r_4.$$

2. Цикл, формирующий содержимое адреса r_1 (приведенный индекс элемента a_{ij}), не зависит от параметра k, и поэтому может быть вынесен за соответствующую формулу циклирования. Таким образом, один раз полученный приведенный индекс элемента a_{ij} при данном значении $'j$ будет использоваться в каждом из циклов по $'k$, в результате число тактов работы алгоритма сократится.

Алгоритм *Прямой ход* преобразуется к виду

Прямой ход 4... $\boldsymbol{Ц}\{2\,(1)\,n \Rightarrow i\}\,M4$
$$\boldsymbol{Ц}\{'i\,(1)\,n \oplus 1 \Rightarrow j\}\,M4$$
$$'j \Rightarrow r_1$$
$$\boldsymbol{Ц}\{1\,(1)\,'i \ominus 1 \Rightarrow s\}$$
$$'r_1 \oplus (n \ominus 's \oplus 1) \Rightarrow r_1$$
$$\boldsymbol{Ц}\{1\,(1)\,'i \ominus 1 \Rightarrow k\}\,M4$$
$$'i \Rightarrow r_2;\ 'j \Rightarrow r_3;\ 'k \Rightarrow r_4$$
$$\boldsymbol{Ц}\{1\,(1)\,'k \ominus 1 \Rightarrow s\}$$
$$'r_2 \oplus (n \ominus 's \oplus 1) \Rightarrow r_2;\ 'r_3 \oplus (n \ominus 's \oplus 1) \Rightarrow r_3;\ 'r_4 \oplus$$
$$\oplus (n \ominus 's \oplus 1) \Rightarrow r_4$$
$$'(\alpha \oplus 'r_1) - \frac{'(\alpha \oplus 'r_2) \cdot '(\alpha \oplus 'r_3)}{'(\alpha \oplus 'r_4)} \Rightarrow \alpha \oplus 'r_1$$
$$M4\ \ldots\ Я.$$

3. Рассмотрим вторую и третью по порядку формулы циклирования. В области действия третьей формулы циклирования содержится лишь один адрес r_1, зависящий от $'j$. В связи с этим преобразуем этот цикл к виду, не зависящему от параметра $'j$, учитывая, что три строки алгоритма

$$'j \Rightarrow r_1$$
$$\textbf{Ц}\{1\,(1)'i \ominus 1 \Rightarrow s\}$$
$$'r_1 \oplus (n \ominus 's \oplus 1) \Rightarrow r_1$$

эквивалентны следующим четырем:

$$0 \Rightarrow r$$
$$\textbf{Ц}\{1\,(1)'i \ominus 1 \Rightarrow s\}$$
$$'r \oplus (n \ominus 's \oplus 1) \Rightarrow r$$
$$'r \oplus 'j \Rightarrow r_1,$$

из которых первые три определяют цикл, не зависящий от $'j$. Теперь первые три из этих строк могут быть вынесены за вторую формулу циклирования. Это также дает экономию в тактах работы алгоритма. Таким образом, вместо первых пяти строк следует записать следующие строки:

Прямой ход 4а... $\textbf{Ц}\{2\,(1)\,n \Rightarrow i\}\,M4$
$$0 \Rightarrow r$$
$$\textbf{Ц}\{1\,(1)'i \ominus 1 \Rightarrow s\}$$
$$'r \oplus (n \ominus 's \oplus 1) \Rightarrow r$$
$$\textbf{Ц}\{'i\,(1)\,n \oplus 1 \Rightarrow j\}\,M4$$
$$'r \oplus 'j \Rightarrow r_1$$
$$\ldots$$

4. Рассмотрим первые четыре строки последнего алгоритма. Здесь адрес r играет роль счетчика, однако суммируемые величины $n \ominus 's \oplus 1$ определяются параметром $'s$ (счетчик с формируемым шагом). Если адрес r в остальных строках алгоритма не используется, то работу этого счетчика можно организовать так, чтобы в каждом последующем цикле по $'i$ было использовано его содержимое, полученное на предыдущем цикле, а не вычислялось каждый раз сначала, как это было в программе *Прямой ход 4а.*

Построение такого счетчика с накоплением осуществляется перестановкой двух первых формул алгоритма *Прямой ход 4А*, выбрасыванием третьей строки (формулы циклирования по $'s$) и заменой в четвертой строке адреса $'s$ на $'i - 1$ (т. е. на последнее из значений $'s$). Таким образом, получаем программу:

$$\textit{Прямой ход 4б}\ldots\ 0 \Rightarrow r$$
$$Ц\{2(1)\, n \to i\}\, M4$$
$$'r \oplus (n \ominus 'i \oplus 2) \to r$$

6. Преобразования, аналогичные предыдущим, могут быть проведены в цикле по параметру k.

Строки алгоритма

$$Ц\{1(1)'i \ominus 1 \to k\}\, M4$$
$$'i \to r_2;\ 'j \to r_3;\ 'k \Rightarrow r_4$$
$$Ц\{1(1)'k \ominus 1 \Rightarrow s\}\, Q$$
$$'r_3 \oplus (n \ominus 's \oplus 1) \to r_2;\ 'r_3 \oplus (n \ominus 's \oplus 1) \Rightarrow r_3$$
$$'r_4 \oplus (n \ominus 's \oplus 1) \Rightarrow r_4$$
$$Q\ldots$$
$$\ldots\ldots$$

заменим сперва, как и в п. 3, на эквивалентные им следующие:

$$Ц\{1(1)'i \ominus 1 \to k\}\, M4$$
$$0 \Rightarrow t$$
$$Ц\{1(1)'k \ominus 1 \Rightarrow s\}$$
$$'t \oplus (n \ominus 's \oplus 1) \Rightarrow t$$
$$'t \oplus 'i \Rightarrow r_2;\ 't \oplus 'j \Rightarrow r_3;\ 't \oplus 'k \Rightarrow r_4.$$

(Метка Q не нужна, так как теперь внутренний цикл распространяется лишь на одну строку).

6. Попытаемся, как и в п. 4, выбросить формулу циклирования по s. В отличие от рассмотренного в п. 4 случая цикл по $'s$ при $'k = 1$ равносилен пустой строке и поэтому в этом случае аналогичное преобразование неосуществимо.

Выполним эквивалентное преобразование первых трех строк алгоритма, в результате которого внутренний цикл не будет обращаться в пустую строку, учитывая, что

первые четыре строки последнего алгоритма эквивалентны следующим строкам:

$$\begin{gathered}\boldsymbol{Ц}\{1\,(1)'\,i \ominus 1 \Rightarrow k\}\,M4\\ -(n \oplus 1) \Rightarrow t\\ \boldsymbol{Ц}\{1\,(1)'\,k \Rightarrow s\}\\ t' \oplus (n \ominus 's \oplus 2) \Rightarrow t.\end{gathered}$$

Выполнив преобразование, аналогичное описанному в п. 4, получим:

$$\begin{gathered}-(n+1) \Rightarrow t\\ \boldsymbol{Ц}\{1\,(1)'\,i - 1 \Rightarrow k\}\,M4\\ 't \oplus (n \ominus 'k \oplus 2) \Rightarrow t.\end{gathered}$$

Для устранения зависимости алгоритма от параметров α и n примем

$$'\varphi = \alpha,\ ''\varphi = n.$$

Окончательно запишем рассматриваемый алгоритм в виде

$$\begin{gathered}\textit{Прямой ход}\ \ldots\ '\varphi \Rightarrow r\\ \boldsymbol{Ц}\{2\,(1)''\varphi \Rightarrow i\}\,M4,\ Я\\ 'r \oplus (''\varphi \ominus 'i \oplus 2) \Rightarrow r\\ \boldsymbol{Ц}\{'i\,(1)''\varphi \oplus 1 \Rightarrow j\}\,M4\\ 'r \oplus 'j \Rightarrow r_1\\ -(n \oplus 1) \Rightarrow t\\ \boldsymbol{Ц}\{1\,(1)'\,i \ominus 1 \Rightarrow k\}\,M4\\ 't \oplus (''\varphi \ominus 'k \oplus 2) \Rightarrow t\\ 't \oplus 'i \Rightarrow r_2;\ 't \oplus 'j \Rightarrow r_3;\ 't \oplus 'k \Rightarrow r_4\\ '('\varphi \oplus 'r_1) - \frac{'('\varphi \oplus 'r_2) \cdot '('\varphi \oplus 'r_3)}{'('\varphi \oplus 'r_4)} \Rightarrow '\varphi \oplus 'r_1\\ M4\ldots\end{gathered} \tag{3.59}$$

Вычисление приведенных индексов элементов по формуле (3.56) в приведенной программе осуществляется рекурсивно: при заданном $'i$ по адресу r хранится значение суммы

$$\Sigma = \sum_{k=1}^{i-1} (n - k + 1),$$

которое затем используется при увеличении $'i$ на единицу; аналогично по адресу t при заданном $'k$ хранится значение соответствующей суммы.

Эти суммы можно вычислить по формуле суммы арифметической прогрессии, как это было сделано в предыдущем параграфе, в результате чего была получена программа (3.54).

Однако по числу выполняемых операций программа (3.59) экономнее. Действительно, для вычисления нового значения Σ нужно к предыдущему ее значению прибавить $''\varphi - i + 2$, т. е. выполнить две операции; для вычислений по формуле (3.57) таких операций следует выполнить 5.

Пример 2. Рассмотрим алгоритм умножения симметричной матрицы $\{a_{ij}\}$, $1 \leqslant i \leqslant n$; $1 \leqslant j \leqslant n$; $'a_{ij} = 'a_{ji}$, на вектор, с компонентами $'b_j$, $1 \leqslant j \leqslant n$. Для компонент результирующего вектора отведем множество адресов c_i, $1 \leqslant i \leqslant n$.

Расчетные формулы при этом имеют вид

$$'c_i = \sum_{j=1}^{n}{}' a_{ij} \cdot 'b_j \qquad (i = 1,2, \ldots, n). \tag{3.60}$$

Учитывая, что $'a_{ij} = 'a_{ji}$, можем считать заданным лишь множество адресов a_{ij}, для которых $j \geqslant i$. В этом случае расчетные формулы принимают вид

$$'c_i = \sum_{j=1}^{i}{}' a_{ji} \cdot 'b_j + \sum_{j=i+1}^{n}{}' a_{ij} \cdot 'b_j \qquad (i = 1, 2, \ldots, n). \tag{3.60}$$

Предполагая, что вначале $'c_i = 0$ $(i = 1,2, \ldots, n)$, программу вычислений по формулам (3.60) на общеалгоритмическом уровне можем записать в виде

$$\begin{gathered} \textit{симм}\ 1\ldots\ \boldsymbol{Ц}\{1\,(1)\,n \Rightarrow i\},\ \textit{Я} \\ \boldsymbol{Ц}\{1\,(1)\,n \Rightarrow j\} \\ 'c_i + 'a_{ij} \times 'b_j \Rightarrow c_i. \end{gathered}$$

Соответственно для формул (3.61) имеем программу

$$\begin{gathered} \textit{симм}\ 2\ldots\ \boldsymbol{Ц}\{1\,(1)\,n \Rightarrow i\}\ M,\ \textit{Я} \\ \boldsymbol{Ц}\{1\,(1)'\,i \Rightarrow j\} \\ 'c_i + 'a_{ji} \cdot 'b_j \Rightarrow c_i \\ \boldsymbol{Ц}\{'i \oplus 1\,(1)\,n \Rightarrow j\} \\ 'c_i + 'a_{ij} \cdot 'b_j \Rightarrow c_i \\ M\ldots \end{gathered} \tag{3.62}$$

или

$$\begin{gathered}\boldsymbol{Ц}\{1\,(1)\,n \Rightarrow i\},\ \boldsymbol{Я}\\ \boldsymbol{Ц}\{1\,(1)\,n \Rightarrow j\}\\ \boldsymbol{P}\{'i \leqslant 'j\}\,'a_{ij} \times 'b_j + 'c_i \Rightarrow c_i \downarrow 'a_{ji} \times 'b_j + 'c_i \Rightarrow c_i.\end{gathered} \tag{3.63}$$

Программы (3.62) и (3.63) основаны на схемах обозревания информации по индексу i — номеру элемента результирующего вектора: циклы по параметру j осуществляют обозревание всех элементов информации, необходимых для формирования содержимого адреса c_i.

Обозревание информации в данном примере можно осуществить по элементам исходной матрицы: элементы исходной матрицы обозреваются построчно и однократно.

При этом в зависимости от отношения $'i = 'j$ (т. е. от того, является элемент диагональным или нет) необходимо выполнить засылки

$$'a_{ij} \times 'b_j + 'c_i \Rightarrow c_i \qquad (i = 1,\ 2,\ \ldots,\ n)$$

при $'i = 'j$ и

$$\left.\begin{matrix}'a_{ij} \times 'b_j + 'c_i \Rightarrow c_i\\ 'a_{ij} \times 'b_i + 'c_j \Rightarrow c_j\end{matrix}\right\}\ (j > i;\ 1 \leqslant i \leqslant n)$$

при $'i \neq 'j$.

В этом случае получаем программу

$$\begin{gathered}\textit{симм}\ 3\ldots\ \boldsymbol{Ц}\{1\,(1)\,n \Rightarrow i\}\,M,\ \boldsymbol{Я}\\ \boldsymbol{Ц}\{'i\,(1)\,n \Rightarrow j\}\,M\\ \boldsymbol{P}\{'i = 'j\}\,M1\\ 'a_{ij} \times 'b_j + 'c_i \Rightarrow c_i\\ M1\ldots\ 'a_{ij} \times 'b_i + 'c_j \Rightarrow c_j\\ M\ldots\end{gathered}$$

Запишем алгоритм, реализованный в программе *симм 3* на уровне условных адресов, полагая, что симметричная матрица задана по строкам своими диагональными и наддиагональными элементами в виде α-последовательности, компоненты вектора заданы в виде β-последовательности, а компоненты результирующего вектора требуется получить в виде γ-последовательности.

Учитывая, что

$$\left.\begin{matrix}a_{ij} = '\left(\alpha \oplus 'j \oplus \left(n \ominus \frac{'i}{2}\right) \otimes ('i \ominus 1)\right);\\ b_i = '(\beta \oplus 'i);\\ \gamma_i = '(\gamma \oplus 'i)\end{matrix}\right\}\ (i = 1,\ 2,\ \ldots\ n;\ j \geqslant i)$$

получаем программу в виде

$$\textit{симм } 4\ldots \quad \boldsymbol{Ц}\{1\,(1)\,n \Rightarrow i\}\,M,\ \textit{Я}$$
$$\boldsymbol{Ц}\{'i\,(1)\,n \Rightarrow j\}\,M$$
$$\boldsymbol{P}\{'i = 'j\}\,M1$$
$$'\left(\alpha \oplus 'j \oplus \left(n \ominus \frac{'i}{2}\right) \otimes (i' \ominus 1)\right) \times '(\beta \oplus 'j) + '(\gamma \oplus 'i) \Rightarrow \gamma \oplus 'i$$
$$M1\ldots \quad '\left(\alpha \oplus 'j \oplus \left(n \ominus \frac{'i}{2}\right) \otimes ('i \ominus 1)\right) \times '(\beta \oplus 'i) +$$
$$+ '(\gamma \oplus j) \Rightarrow \gamma \oplus 'j$$
$$M\ldots$$

Вынося вычисления, не зависящие от индекса j, за формулу циклирования по j (это дает экономию в числе тактов работы алгоритма), получим

$$\textit{симм } 5\ldots \quad \boldsymbol{Ц}\{1\,(1)\,n \Rightarrow i\}\,M,\ \textit{Я}$$
$$\alpha \oplus \left(n \ominus \frac{'i}{2} \otimes ('i \ominus 1) \Rightarrow s;\ '(\gamma \oplus 'i) \Rightarrow t;\ '(\beta \oplus 'i) \Rightarrow r\right.$$
$$\boldsymbol{Ц}\{'i\,(1)\,n \Rightarrow j\}\,M$$
$$\boldsymbol{P}\{'i = 'j\}\,M1$$
$$'('s \oplus 'j) \times '(\beta \oplus 'j) + 't \Rightarrow t$$
$$M1\ldots \quad '('s \oplus 'j) \times 'r + '(\gamma \oplus 'j) \Rightarrow \gamma \oplus 'j$$
$$M\ldots$$

ГЛАВА IV

АДРЕСНОЕ ПРОГРАММИРОВАНИЕ И МЕТОД БИБЛИОТЕЧНЫХ ПОДПРОГРАММ

1. ПРИНЦИП БИБЛИОТЕЧНЫХ ПОДПРОГРАММ

В алгоритмах различных задач могут встретиться формально одинаковые части, либо такие части могут быть получены с помощью формул замены. Благодаря этому программы сложных задач можно строить из программ более простых задач, называемых подпрограммами. Такой метод автоматизации программирования называется методом библиотечных подпрограмм [14].

С помощью формулы вхождения $\Pi\beta\{\alpha_1, \ldots, \alpha_n\}\gamma$ любая программа в адресном языке может быть сделана библиотечной подпрограммой. Для этого достаточно ее начальную строку сделать строкой засылок или замен с пустыми левыми частями и пометить меткой β, обозначающей библиотечный код подпрограммы, и заканчивать подпрограмму формулой останова $Я$. Тогда обращение к данной библиотечной подпрограмме с последующим возвратом к строке с меткой γ осуществляется формулой вхождения $\Pi\beta\{\ldots\}\gamma$. Эта формула равносильна записи целой подпрограммы с библиотечным номером β, в которой символ $Я$ заменен на γ.

Таким образом, любой алгоритм, включенный в библиотеку подпрограмм, фактически превращается в элементарную операцию. Это упрощает программирование уже на уровне адресного языка.

Еще более важен принцип библиотечных подпрограмм для конкретных машин, имеющих, в отличие от ИАВМ, ограниченный (хотя и полный) набор операций. Библиотечные подпрограммы рационально составлять и для действий, которые в ИАВМ являются элементарными и записываются в одну строку адресного алгоритма. Такие подпрограммы являются программами моделирования

операций НАВМ на данной машине. Простейшие примеры обращения к подпрограммам были приведены в гл. I (примеры 4 и 3).

Подпрограммы могут составляться с учетом возможности изменения вспомогательных параметров, таких как размерности векторов, таблиц, порядок интегрируемой системы дифференциальных уравнений и т. п. Поэтому подпрограммам, помимо непосредственно исходных данных, должны задаваться еще значения соответствующих параметров, в связи с чем число аргументов в подпрограммах может оказаться весьма значительным.

Применяя стандартные способы кодирования и схемы обозревания информации по адресам высших рангов, можно передавать все необходимые данные для подпрограмм с помощью всего лишь одного-двух адресов, хранящих информацию о распределении этих данных по адресам. Поясним сказанное примерами.

1. Пусть исходными данными подпрограммы служат компоненты вектора $a_1, a_2, \ldots, a_n$, размещенные в последовательности адресов

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha \oplus n.$$

По адресу α поместим информацию о размерности вектора $'\alpha = n$. Вся информация о полученной последовательности α, $\alpha \oplus 1$, $\alpha \oplus 2$, ..., $\alpha \oplus {}'\alpha$ передается стандартной подпрограмме с помощью засылки ведущего адреса α в стандартный адрес подпрограммы f:

$$\alpha \Rightarrow f.$$

Таким образом, информация о векторе задается подпрограмме в виде содержимого адреса f:

$$'f = \alpha;\ {}^2f = n;\ '('f \oplus i) = '(\alpha \oplus i) = a_i.$$

2. Пусть элементы последовательности (компоненты вектора) кодируются в последовательность адресов по k в один адрес так, что каждому из элементов, размещенных по адресу φ, отводится элементарный адрес $(\varphi)_s$ (где $s = 1, 2, \ldots, k$). По адресу α задается число n элементов в последовательности

$$'\alpha = n;\ \alpha \Rightarrow f.$$

Тогда

$$'f = \alpha;\ ''f = n'\left(f \oplus \text{ц. ч. } \frac{i}{k} \oplus 1\right)_{\text{д. ч.}\frac{i}{k}k} = a_i,$$

где ц.ч. и д.ч. обозначают соответственно целую и дробную часть $\frac{i}{k}$. Выделению содержимого элементарного адреса $'\varphi_s$ соответствуют следующие действия

$$'\varphi_s = ('\varphi \wedge 'cs)\, сдв\, 'ds,$$

где $\wedge$ — символ поразрядной операции дизъюнкции,

$'cs$ — константа, содержащая единицы в разрядах, соответствующих s-му элементарному адресу и нули в остальных разрядах (константа выделения);

$сдв$ — символ операции сдвига,

$'ds$ — константа, определяющая сдвиг содержимого s-го адреса в позицию, в которой это содержимое приобретает свой естественный, например, арифметический смысл (константа сдвига).

В некоторых случаях в сдвиге нет необходимости и $'\varphi_s = '\varphi \wedge 'cs$.

Засылке $'a_k$ по элементарному адресу φ_s (при условии, что адреса a_k и φ_s содержат парное число разрядов, соответствует действие

$$(('a \wedge 'ck)\, сдв\, ('ds - 'dk)) \vee ('\varphi_s \wedge 'нcs) \Rightarrow \varphi.$$

Здесь $'нcs$ — константа, содержащая нули в разрядах, соответствующих s-му адресу, и единицы в остальных разрядах (константа гашения); $\vee$ — символ поразрядной операции дизъюнкции; $'ck$, $'ds$, $'dk$ — соответствующие константы выделения и сдвига.

3. Информация о квадратной матрице может быть задана также с помощью одного адреса. Отведем компонентам матрицы последовательность адресов $\alpha + 1$, $\alpha + 2$, ..., $\alpha + n^2$, предполагая их перебор, например, по строкам. В ячейку α поместим размерность матрицы n. Информация о последовательности

$$\alpha,\ \alpha \oplus 1,\ \ldots,\ \alpha \oplus ('\alpha)^2$$

стандартной программе может быть задана с помощью засылки в ее стандартный адрес f адреса α:

$$\alpha \Rightarrow f.$$

Тогда по адресу f может быть найдена размерность матрицы

$$'f = \alpha;\ {}^2f = n,$$

а также адрес любого элемента матрицы

$$a_{ij} = '('f \oplus (i \ominus 1)\,({}^2f) \oplus j).$$

В случае перебора элементов матрицы по столбцам

$$a_{ij} = '('f \oplus (j \ominus 1)\,({}^2f) \oplus i).$$

4. Аналогично может задаваться информация о прямоугольной матрице с m столбцами и n строками. Разместим по адресам $\alpha \oplus 2,\ \alpha \oplus 3,\ \ldots,\ \alpha \oplus mn \oplus 1$ компоненты матрицы по строкам; по адресу $\alpha \oplus 1$ поместим число m, по адресу α — число n. Получим последовательность $\alpha,\ \alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ '(\alpha \oplus 1)'\alpha \oplus 1$. Как и в предыдущих случаях, подпрограмме, обрабатывающей элементы матрицы, достаточно по ее стандартному адресу f заслать адрес α:

$$\alpha \Rightarrow f,$$

тогда

$$'f = \alpha;\ '('f \oplus 1) = m;$$
$${}^2f = n;\ '('f \oplus (i \ominus 1)\,n \oplus j \oplus 1) = a_{ij}.$$

5. Для симметричной матрицы n-го порядка отводятся адреса лишь для ее диагональных и наддиагональных элементов (в целях экономии памяти). Поместив по адресу α порядок матрицы, а в последовательность адресов

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha \oplus \frac{'\alpha\,('\alpha \oplus 1)}{2}$$

— ее элементы по строкам, стандартной подпрограмме всю информацию о симметричной матрице можно передать одним ее адресом α:

$$\alpha \Rightarrow f.$$

Тогда

$$'f = \alpha;\ {}^2f = n;\ a_{ij} = '\Big(\alpha \oplus \sum_{k=1}^{i-1} (n \ominus k) \oplus j\Big) =$$

$$= '\Big(\alpha \oplus j \oplus \Big(n \ominus \frac{i}{2}\Big)\,(i \ominus 1)\Big).$$

Аналогично для симметричной матрицы, заданной по столбцам,

$$a_{ij} = {}'\left(\alpha \bigoplus i \bigoplus \sum_{k=1}^{j-1} k\right) =$$

$$= {}'\left(\alpha \bigoplus i \bigoplus \frac{j}{2}(j \ominus 1)\right).$$

6. Для матрицы с неполным заполнением могут отводиться адреса лишь для отличных от нуля элементов. Пусть N — число элементов матрицы (${}'\alpha = N$), отличных от нуля; ненулевым элементам матрицы отведены в произвольном порядке адреса

$$\alpha \bigoplus 1,\ \alpha \bigoplus 2,\ \ldots,\ \alpha \bigoplus {}'\alpha.$$

По адресу $\lambda(\alpha \bigoplus {}'\alpha \bigoplus k)$ содержится индекс i (номер строки), а по адресу $\mu(\alpha \bigoplus {}'\alpha \bigoplus k)$ — индекс j (номер столбца) элемента, содержащегося по адресу $\alpha \bigoplus k$. Здесь через λ и μ обозначены некоторые функции. Например, $\lambda(s) = s_{\mathrm{I}}$ и $\mu(s) = s_{\mathrm{II}}$, где s_{I} и s_{II} обозначают элементарные адреса (части адреса).

Тогда, полагая, что $\alpha \Rightarrow f$, получаем

$${}'f = \alpha,\ {}''f = N,$$

и, если

$$a_{ij} = {}'({}'f \bigoplus k),\quad 1 \leqslant k \leqslant {}^{2}f,$$

то

$${}'\mu({}'f \bigoplus {}''f \bigoplus k) = j;\ {}'\lambda({}'f \bigoplus {}''f \bigoplus k) = i.$$

Этот способ кодирования удобен и в случае многомерных матриц с неполным заполнением.

7. Для прямоугольной матрицы, элементы которой a_{ij} являются векторами с компонентами $a_{ij}(k)$, $k = 1, 2, \ldots n_{ij}$ ($i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$), можно принять:

${}'\alpha = n$ — число строк матрицы;

${}'(\alpha \bigoplus 1) = m$ — число столбцов матрицы;

${}'(\alpha \bigoplus 2) = n_{11}$ — размерность вектора a_{11};

${}'(\alpha \bigoplus 3) = n_{12}$ — размерность вектора a_{12};

. . . . . .

${}'(\alpha \bigoplus mn) = n_{mn}$ — размерность вектора a_{mn};

${}'(\alpha \bigoplus mn \bigoplus 1) = a_{11}(1)$ — первая компонента вектора a_{11} и т. д.

Тогда, принимая $\alpha \Rightarrow f$, получаем:

$$'f = \alpha;\ ''f = n$$
$$'('f \oplus 1) = m;$$
$$a_{ij}(k) = '(f \oplus 1 \oplus ''f\,'('f \oplus 1) \oplus \sum_{s=2}^{(i-1)''f+j} '('f \oplus s) \oplus k).$$

8. Если положить в предыдущем примере, что размерности всех векторов равны, то формула определения адреса по индексам элемента упрощается. Для прямоугольной матрицы размерности (n, m) с элементами — векторами размерности s (трехмерная матрица) можно принять

$$'\alpha = m;\ '(\alpha \oplus 1) = n;\ '(\alpha \oplus 2) = s$$
$$'(\alpha \oplus 3) = a_{11}(1),\ '(\alpha \oplus 4) = a_{11}(2) \text{ и т. д.}$$

Тогда, если $\alpha \Rightarrow f$, получим

$$'f = \alpha;\ ''f = n;\ '('f \oplus 1) = m;\ '('f \oplus 2) = s$$
$$a_{ij}(k) = 'f \oplus 2 \oplus ((i \ominus 1)\,''f \oplus j)\,s \oplus k.$$

9. В частном случае при вычислении с помощью подпрограмм значений функций одного или двух переменных в стандартные адреса подпрограммы f, $f1$ засылаются непосредственно аргументы этих функций.

10. Если подпрограмма обрабатывает и выдает несколько массивов, то может быть построена объединяющая α-последовательность. Способы построения такой последовательности поясним на примере подпрограммы умножения двух матриц.

Пусть матрица A задана $\alpha 1$-последовательностью, матрица B — $\alpha 2$-последовательностью и матрица $C = A \times B$ должна быть получена в виде $\alpha 3$-последовательности (все последовательности имеют некоторую стандартную структуру).

Построим α-последовательность, содержащую ведущие адреса указанных последовательностей:

$$'\alpha = \alpha 1$$
$$'(\alpha \oplus 1) = \alpha 2$$
$$'(\alpha \oplus 2) = \alpha 3.$$

Для обращения к подпрограмме необходимо выполнить засылку $\alpha \Rightarrow f$. Тогда

$$''f = \alpha 1;\ '('f \oplus 1) = \alpha 2;\ '('f \oplus 2) = \alpha 3;$$

$'''f = n$ и т. д. (где n — размерность матрицы A). В этом случае ранг адреса, с помощью которого обозреваются элементы входной информации, повысился до трех.

Ни в одном из этих примеров явно не участвуют адреса, содержащие элементы информации. Таким образом, информация для стандартной программы не связывается с местом размещения обрабатываемой и формируемой информации (которую назовем внешней) и, если подпрограмму в задаче нужно применять многократно, достаточно лишь подготавливать содержимое адреса f.

В некоторых случаях можно не объединять всю информацию в единую α-последовательность, если ввести понятие (k, l) — местности подпрограммы по количеству k-обрабатываемых и l-выдаваемых массивов. Для (k, l) — местной подпрограммы — выделяется $k + l$ ячеек, в которых содержится внешняя информация. Если результат программы — массив, то он выдается тоже в виде стандартной α-последовательности. Это удобно при переходе от подпрограммы к подпрограмме.

Таким образом, при стандартном способе кодирования информации для большинства подпрограмм (принципиально для всех) достаточно одного-двух стандартных адресов, по которым может быть передана информация о массивах исходных и выходных данных.

2. ПРОГРАММИРОВАНИЕ ЗАДАЧ ЛИНЕЙНОЙ АЛГЕБРЫ

Схемами обозревания информации, характерными для задач линейной алгебры, в значительной степени исчерпываются способы обозревания информации в самых разнообразных задачах. К программам этих задач предъявляются следующие требования.

1. В задачах линейной алгебры входными и выходными данными могут служить скалярные величины, векторы или матрицы, а также матрицы, элементами которых являются некоторые векторы. В случае векторных и матричных величин предполагаем, что их компоненты размещены в α-последовательностях, примеры которых проводились ранее.

2. Рабочие адреса, т. е. адреса, используемые лишь для работы подпрограммы, могут быть общими для всех стандартных подпрограмм. (Это требование не является принципиальным, оно введено с целью экономии адресов). В частности, принимаем обозначения для адресов — носителей выходной информации: $\psi 1, \psi 2, \ldots$; для адресов — носителей выходной информации: $\varphi 1, \varphi 2, \ldots$

Пример 1. *Стандартная подпрограмма вычитания n-мерных векторов:*

$$X - Y = Z.$$

Пусть вектор X задан в виде α-последовательности, т. е.

$$'\alpha = n, \ '(\alpha \oplus i) = x_i, \ (i = 1, 2, \ldots, n),$$

где x_i — компоненты вектора, а вектор Y — в виде β-последовательности

$$\beta, \ \beta \oplus 1, \ \ldots, \ \beta \oplus '\beta.$$

Вектор-разность Z необходимо получить в виде γ-последовательности.

Обращение к подпрограмме осуществляется по формуле

$$\boldsymbol{\Pi} B1 \ \{\alpha, \beta, \gamma\},$$

а сама подпрограмма записывается в виде

$$\begin{gathered} B1 \ \ldots \ \varnothing \twoheadrightarrow \psi 1, \ \varnothing \twoheadrightarrow \psi 2, \ \varnothing \twoheadrightarrow \varphi 1 \\ \boldsymbol{Ц} \ \{1 \ (1)^2 \psi 1 \twoheadrightarrow \delta\} \\ '('\psi 1 \oplus '\delta) — '('\psi 2 \oplus '\delta) \twoheadrightarrow '\varphi 1 \oplus '\delta \\ {}^2\psi 1 \twoheadrightarrow '\varphi 1 \\ \boldsymbol{Я}. \end{gathered} \tag{4.1}$$

Здесь δ играет роль некоторого счетчика, работающего по принципу сложения.

Так как ${}^2\psi 1 = n$ — размерности векторов, то в результате n-кратного выполнения второй строки программы в адресах $\gamma \oplus 1, \ldots, \gamma \oplus n$ получаются компоненты вектора Z. Четвертая строка в программе необходима для задания размерности результирующего вектора Z по адресу $\gamma = '\varphi 1$.

Эта же программа, в случае счетчика δ, работающего по принципу вычитания, записывается в виде

$$\begin{gathered} B_1 \ \ldots \ \varnothing \twoheadrightarrow \psi 1, \ \varnothing \twoheadrightarrow \psi 2, \ \varnothing \twoheadrightarrow \varphi 1 \\ \boldsymbol{Ц} \ \{{}^2\psi 1 \ (\ominus 1) \ 1 \twoheadrightarrow \delta\} \\ '('\psi 1 \oplus {}^2\psi 1 \ominus '\delta) — '('\psi 2 \oplus {}^2\psi 2 \ominus '\delta) \twoheadrightarrow '\varphi 1 \oplus {}^2\psi \ominus '\delta \\ {}^2\psi 1 \twoheadrightarrow '\varphi 1 \\ \boldsymbol{Я} \end{gathered}$$

или в виде

$$B_1 \ldots \varnothing \to \psi 1,\ \varnothing \to \psi 2,\ \varnothing \to \varphi 1$$
$$Ц\ \{('\psi 1 \oplus 1)\,(1)\,('\psi 1 \oplus {}^2\psi 1) \to \delta 1;\ ('\psi 2 \oplus 1)\,(1) \to \delta 2;$$
$$('\varphi 1 \oplus 1)\,(1) \to \delta 3\}$$
$${}^2\delta 1 - {}^2\delta 2 \to '\delta 3$$
$${}^2\psi 1 \to '\varphi 1$$
$$Я.$$

В дальнейшем из-за её громоздкости подобной записью пользоваться не будем.

Пример 2. *Стандартная подпрограмма умножения вектора на скаляр:*

$$Xa = Y.$$

Пусть вектор X задан в виде α-последовательности; скаляр a помещен по адресу β : $'\beta = a$, а результирующий вектор Y необходимо получить в виде γ-последовательности.

Полагая, что обращение к подпрограмме осуществляется по формуле

$$ПВ2\ \{\alpha,\ '\beta,\ \gamma\},$$

получаем программу. Учитываем, что по адресу $\psi 2$ засылается содержимое адреса β, т. е. скаляр a, тогда как по адресам $\psi 1$ и $\varphi 1$ засылаются сами адреса α и γ. Это объясняется тем, что $\psi 1$ и $\varphi 1$ в программе используются как адреса второго ранга

$$B2 \ldots \varnothing \to \psi 1,\ \varnothing \to \psi 2,\ \varnothing \to \varphi 1$$
$$Ц\ \{1\,(1)^2\,\psi 1 \to \delta\}$$
$$'('\psi 1 + '\delta) \times '\psi 2 \to '\varphi 1 + '\delta$$
$${}^2\psi 1 \to '\varphi 1$$
$$Я$$

или

$$B2 \ldots \varnothing \to \psi 1,\ \varnothing \to \psi 2,\ \varnothing \to \varphi 1$$
$$Ц\ \{{}^2\psi 1\,(\ominus 1)\,1 \to \delta\}$$
$$'('\psi 1 \oplus {}^2\psi 1 \ominus '\delta) \times '\psi 2 \to '\varphi 1 \oplus {}^2\psi 1 \ominus '\delta$$
$${}^2\psi 1 \to '\varphi 1$$
$$Я.$$

Пример 3. *Стандартная подпрограмма вычисления квадрата модуля вектора:*

$$\| X \|^2 = a.$$

Пусть вектор X задан в виде α-последовательности; результат (скалярная величина) должен быть получен по адресу β.

Полагая, что обращение к подпрограмме выполняется по формуле

$$ПВ3\ \{\alpha,\ \beta\},$$

получаем программу

$$B3 \ldots \varnothing \to \psi 1,\ \varnothing \to \varphi 1,$$
$$0 \to '\varphi 1$$
$$Ц\ \{1\,(1)^2\,\psi 1 \to \delta\},\ Я$$
$$['('\psi 1 \oplus '\delta)]^2 + {}^2\varphi 1 \to '\varphi 1.$$

Пример 4. *Стандартная подпрограмма умножения квадратной матрицы A на вектор X:*

$$AX = Y.$$

Пусть матрица A задана по строкам в виде α-последовательности, т. е. $'\alpha = n$ (n — размерность матрицы), $a_{ij} = '(\alpha \oplus (i \ominus \ominus 1)\,'\alpha \oplus j)$; вектор X — в виде β-последовательности. Вектор Y необходимо получить в виде γ-последовательности.

Полагая, что обращение к подпрограмме выполняется по формуле

$$\Pi B4\ \{\alpha,\ \beta,\ \gamma\}$$

и

$$'(\gamma \oplus i) = 0 \quad (i = 1,\ 2,\ \ldots,\ n),$$

получаем программу

$$B_4\ \ldots\ \varnothing \to \psi 1,\ \varnothing \to \psi 2,\ \varnothing \to \varphi 1$$
$$Ц\ \{1\ (1)^2\ \psi 1 \to \delta 1\}$$
$$Ц\ \{1\ (1)^2\ \psi 2 \to \delta 2\}$$
$$'('\psi 1 \oplus ('\delta 1 \ominus 1)^2\,\psi 1 \oplus '\delta 2) \cdot '('\psi 2 \oplus '\delta 2) + '('\varphi 1 \oplus '\delta 1) \to '\varphi 1 \oplus '\delta 1$$
$$^2\psi 1 \to '\varphi 1$$
$$Я.$$

Рекомендуем читателю составить программу для случая работы счетчиков по принципу вычитания.

Пример 5. *Составление стандартной программы умножения матрицы, транспонированной к заданной матрице A, на вектор X (предполагается заданной по строкам матрица A):*

$$A^*X = Y.$$

Воспользовавшись обозначениями предыдущего примера, получим

$$B5\ \ldots\ \varnothing \to \psi 1,\ \varnothing \to \psi 2,\ \varnothing \to \varphi 1$$
$$Ц\ \{1\ (1)^2\ \psi 1 \to \delta 1\}$$
$$Ц\ (1\ (1)^2\ \psi 2 \to \delta 2\}$$
$$'('\psi 1 \oplus '\delta 1 \oplus ('\delta 2 \ominus 1)^2\,\psi 2) \cdot '('\psi 2 \oplus '\delta 2) + '('\varphi 1 \oplus '\delta 1) \to '\varphi 1 \oplus '\delta 1$$
$$^2\psi 2 \to '\varphi 1$$
$$Я.$$

Приведем пример более сложной задачи, решение которой сводится к использованию приведенных подпрограмм.

Пример 6. *Решение системы линейных алгебраических уравнений методом Фридмана.* Система линейных алгебраических уравнений

$$AX = F,$$

(где $A\,(a_{ij})$ — матрица коэффициентов системы; $F\,(f_1,\ f_2,\ \ldots,\ f_n)$ — вектор свободных членов; $X\,(x_1,\ x_2,\ \ldots,\ x_n)$ — искомый вектор решений) может быть решена путем следующего итерационного процесса [18].

Пусть $X^0 = (x_1^0, \ldots, x_n^0)$ — некоторый (произвольный) вектор нулевых приближений. Тогда вектор $(k+1)$-х приближений $X^{(k+1)}$ определяется по формуле

$$X^{(k+1)} = X^{(k)} - a_{k+1} \cdot A^* (AX^{(k)} - F),$$

где

$$a_{k+1} = \frac{\| AX^{(k)} - F \|^2}{\| A^* (AX^{(k)} - F) \|^2}$$

(Здесь $\| F \|$ обозначает модуль вектора F; A^* — транспонированная матрица A). Итерационный процесс прекращается, когда $\| AX^{(k+1)} - F \|^2$ становится меньше некоторого заданного числа ε ($'\omega = \varepsilon$); значение $X^{(k+1)}$ является решением.

Пусть исходная матрица A задана по строкам в виде α-последовательности, вектор правых частей F — в виде ϑ-последовательности, вектор нулевых приближений X^0 — в виде β-последовательности. Вектор решения получаем в этой же последовательности. Для рабочих ячеек, в которые в процессе вычислений помещаются компоненты векторов

$$AX^{(k)},\ AX^{(k)} - F,\ A^* (AX^{(k)} - F),$$

выделяем δ- и δ_1-последовательности. В результате получим программу

```
B6 ... ПB4 {α, β, δ}
       ПB1 {δ, ϑ, δ}
       ПB3 {δ, a}
     Р {'a < 'ω} Я
       ПB5 {α, δ, δ1}
       ПB3 {δ1, b}
       'a : 'b → b
       ПB2 {δ1  b, δ}
       ПB1 {β, δ, β) B6
```

Первая строка означает переход к выполнению программы с входной меткой $B4$ ($AX = Y$). При этом предварительно выполняются засылки:

$$\alpha \rightarrow \psi 1$$
$$\beta \rightarrow \psi 2$$
$$\delta \rightarrow \varphi 1.$$

В результате вектор — произведение исходной матрицы A на вектор нулевых приближений $X^{(0)}$ — получится в виде δ-последовательности. По второй строке подпрограмма с входной меткой $B1$ выдаст вектор

$$AX^{(0)} - F$$

в виде δ-последовательности. Согласно третьей строке алгоритма программа с меткой $B3$ выдает по адресу a квадрат модуля вектора, содержащегося в δ-последовательности. В следующей строке в за-

висимости от величины a совершится переход к следующей строке или на формулу останова $Я$, означающую окончание расчетов. Далее осуществляется переход на подпрограмму с меткой $B5$, которая формирует в δ1-последовательности вектор $A^*(AX^{(k)} - F)$; поскольку вектор, содержащийся в δ-последовательности, должен сохраняться до конца работы этой подпрограммы.

Читателю рекомендуется проследить за ходом дальнейшего выполнения алгоритма.

Превратим полученную программу $B6$ в стандартную подпрограмму. Из набора адресов, выбранных в качестве стандартных для подпрограмм, в подпрограммах $B1$ — $B5$ адреса $\psi 1$, $\psi 2$, $\varphi 1$ уже используются. Поэтому внешнюю информацию к программе $B6$, размещенную в адресах α, ϑ, δ, β, при обращении к этой программе поместим в адреса $\psi 3$, $\psi 4$, $\psi 5$, $\varphi 2$.

Обращение к подпрограмме $B6$ осуществимо по формуле вхождения

$$ПB6\ \{\alpha,\ \vartheta,\ \delta,\ \beta\}\ s.$$

Подпрограмма начнется строкой

$$B6\ \ldots\ \varnothing \to \psi 3,\ \varnothing \to \psi 4,\ \varnothing \to \psi 5,\ \varnothing \to \varphi 2.$$

При этом в программе $B6$ должны быть произведены следующие замены:

символ	α	заменяется на	$'\psi 3$;
»	ϑ	»	» $'\psi 4$;
»	δ	»	» $'\psi 5$;
»	β	»	» $'\varphi 2$.

Пример 7. *Решение систем линейных алгебраических уравнений методом быстрого спуска (метод Качмажа).*

Система линейных алгебраических уравнений вида

$$AX = B.$$

(где $A(a_{ij})$ — матрица коэффициентов; $B(b_1, b_2, \ldots, b_n)$ — вектор свободных членов; $X(x_1, \ldots, x_n)$ — искомый вектор решений, может быть решена следующим образом. Выбирается произвольная точка (начальное приближение) $X^0 = (x_1^0, \ldots, x_n^0)$ и проектируется на гиперплоскость, описываемую первым уравнением; найденная проекция проектируется далее на гиперплоскость, представленную вторым уравнением, затем процесс проектирования на гиперплоскости продолжается в циклическом порядке, причем после каждого шага все ближе и ближе подходим к решению.

Вектор $(k+1)$-ых приближений $X^{(k+1)} = (x_1^{(k+1)}, \ldots, x_n^{(k+1)})$ определяется по формулам

$$x_i^{(k+1)} = x_i^k + l_i a_{ij} \quad (i = 1, 2, \ldots, n),$$

где

$$l_i = \frac{b_i - \sum_{j=1}^{n} a_{ji} x_j}{\sum_{j=1}^{n} a_{ij}^2} \quad (j = 1, 2, \ldots, n).$$

Итерационный процесс прекращается, когда произведение $l_i \cdot a_{ij}$ для всех $j = 1, 2, \ldots, n$ становится меньше некоторого заданного числа ε ($'\omega = \varepsilon$). Значение $X^{(k+1)}$, соответствующее этому произведению, является решением.

Пусть исходная матрица A задана по строкам в виде α-последовательности, вектор правых частей B — в виде β-последовательности, вектор начальных приближений $X^{(0)}$ в виде γ-последовательности. Для рабочих ячеек, в которые в процессе вычислений помещаются компоненты векторов

$$x_i, \quad \sum_{j=1}^{n} a_{ij} \quad (i = 1, 2, \ldots, n),$$

выделяется δ-последовательность размерности $2n + 1$; начало этой последовательности фиксируется в ячейке f ($'f = \delta$). В результате получаем программу

$$B7 \ \ldots \ '\alpha \to 'f$$
$$\textbf{Ц}\,\{1\,(1)\,'\alpha \to i\}\ M2$$
$$\alpha \oplus ('i \ominus 1)\,'\alpha \to a$$
$$\textbf{З}\,\{'\alpha \to {}^{2}\psi 1\}$$
$$\textbf{ПВ}3\,\{'a,\ 'f \oplus '\alpha \oplus 'i\}$$
$$M2 \ \ldots \ \textbf{ПВ}4\,\{\alpha,\ \gamma,\ 'f\}$$
$$\textbf{ПВ}1\,\{\beta,\ 'f,\ 'f\}$$
$$M3 \ \ldots \ \textbf{Ц}\,\{1\,(1)\,'\alpha \to i\}\ M6,\ M2$$
$$'('f \oplus 'i) : '('f \oplus '\alpha \oplus 'i) \to 'f \oplus 'i$$
$$0 \to r1$$
$$M4 \ \ldots \ \textbf{Ц}\,\{1\,(1)\,'\alpha \to j\}\ M5,$$
$$'('f \oplus 'i) \cdot '[\alpha \oplus ('i \ominus 1)\,'\alpha \oplus 'j] \to a$$
$$'(\gamma + 'j) \oplus 'a \to \gamma \oplus 'j$$
$$\textbf{Р}\,\{|\,'a\,| \leqslant '\omega\} \downarrow 1 \to r1$$
$$M5 \ \ldots \ \textbf{Р}\,\{'r_1 = 0\}\ \textit{Я}$$
$$M6 \ \ldots$$

По первым пяти строкам алгоритма для каждого вектора строки матрицы A вычисляется квадрат его модуля по подпрограмме с входной меткой $B3$. В этой подпрограмме требуется, чтобы в начале последовательности, содержащей компоненты вектора, задавалась его размерность. В связи с тем что элементы строк матрицы A записаны сплошным массивом, в программе применяется формула замены (четвертая строка), действие которой распространяется по строку с меткой $M2$. Эта часть программы выполняется один раз.

Согласно строке с меткой $M2$ подпрограмма с входной меткой $B4$ выдает вектор AX, а по следующей строке подпрограмма с меткой $B1$ выдает вектор $B - AX$.

Затем вычисляются значения l_i и для каждого из них вектор $X^{(k+1)}$. Признак того, все ли произведения $(l_i \times a_{ij})$, соответствующие компонентам вектора $X^{(k+1)}$, удовлетворяют условию

$$l_i \times a_{ij} < \varepsilon,$$

ысылается по адресу $r1$. При выполнении последнего условия $0 \to r1$; если, по крайней мере, одно из произведений превышает ε, то $1 \to r1$.

Если найдены проекции на все гиперплоскости, но решение не удовлетворяет требуемой точности, вычисляются новые l_i, и цикл продолжается (начинается проектирование снова на первую гиперплоскость).

Для превращения полученной программы в стандартную подпрограмму необходимо сделать некоторые преобразования. Так как использованные программой подпрограммы B_1, B_3, B_4 содержат адреса ψ_1, ψ_2, φ_1, то в программе 7 должны быть произведены следующие замены:

символ α заменяется на $'\psi_3$;
» β » » $'\psi_4$;
» δ » » $'\psi_5$;
» γ » » $'\varphi_2$.

Подпрограмма начинается строкой

$$B7 \ldots \varnothing \to \psi 3,\ \varnothing \to \psi 4,\ \varnothing \to \psi 5,\ \varnothing \to \varphi 2.$$

Формула вхождения этой подпрограммы будет иметь вид

$$\boldsymbol{П}B7\,\{\alpha, \beta, \delta, \gamma\}.$$

Пример 8. *Вычисление определителя с выбором главного элемента.* Пусть определитель задан α-последовательностью своих элементов по строкам, $'\alpha = n$ — порядку определителя, и по адресу s требуется получить его величину.

Назовем усеченной (α, i)-матрицей $(i = 1, 2, \ldots, n-1)$ совокупность адресов α-последовательности, получаемую в результате выбрасывания из нее всех элементов, соответствующих первым i строкам и первым i столбцам определителя. Максимальный, по модулю, элемент усеченной матрицы назовем главным элементом, а первый ее адрес — главным местом. Введем следующие две подпрограммы.

Подпрограмма *главный элемент*. Эта подпрограмма по заданным α и числу i выдает по адресам ρ и δ соответственно номер строки и столбца максимального по модулю элемента усеченной (α, i)-матрицы, а также переносит главный элемент усеченной матрицы на главное место путем перестановки соответствующей строки и столбца с ее первой строкой и первым столбцом.

Формула вхождения подпрограммы имеет вид

$$\boldsymbol{П}\ \textit{главный элемент}\ (\alpha, i, \rho, \delta).$$

Подпрограмма *нули*. Эта подпрограмма по заданным α и i образует нули под главным местом усеченной (α, i)-матрицы путем подбора соответствующих множителей и последовательного вычитания первой строки из остальных строк. Формула вхождения этой подпрограммы имеет вид

$$\boldsymbol{П}\ \textit{нули}\ (\alpha, i).$$

Выходная информация этой подпрограммы совпадает со входной, т. е. вновь получаемая усеченная матрица кодируется в тех же адресах, в которых размещалась исходная усеченная (α, i)-матрица.

Чтобы сделать программу стандартной, т. е. не зависящей от места размещения определителя, примем $'\varphi = \alpha$. Тогда можем записать следующий алгоритм вычисления определителей:

$$1 \to s$$
$$\textit{Ц}\,\{1\,(1)\,''\varphi \ominus 1 \to \pi\}\,k$$
$$\textit{П}\ \textit{главный элемент}\ \{'\varphi,\ '\pi,\ \rho,\ \delta\}$$
$$'s \times '('\varphi \oplus ('\pi \ominus 1) \bigotimes''\varphi \oplus '\pi) \times (-1)^{'\rho+'\delta} \to s$$
$$\textit{П}\ \textit{нули}\ \{'\varphi,\ '\pi\}$$
$$k\ \ldots\ 's \times '('\varphi \oplus ''\varphi^2) \to s$$
$$\textit{Я},$$

Здесь в четвертой строке происходит умножение главного элемента (с учетом знака) на произведение предыдущих (вначале произведение предыдущих равно 1; это обеспечивается первой строкой алгоритма).

В результате выполнения формулы циклирования происходит выход на строку с меткой k. При этом усеченная $(\alpha,\ n-1)$-матрица состоит из одного (содержащегося по адресу $\alpha + n^2$) элемента, который умножается в этой строке на произведение предыдущих. Тем самым заканчивается процесс вычисления определителя.

Приведем запись подпрограмм, в принципе действия которых рекомендуем разобраться читателю самостоятельно.

Подпрограмма *главный элемент*

$$\textit{главный элемент}\ \ldots\ \varnothing \to \varphi 1,\ \varnothing \to \pi 1,\ \varnothing \to \rho 1,\ \varnothing \to \delta 1$$
$$0 \to \varphi 2$$
$$\textit{Ц}\,\{'\pi 1\,(1)\,''\varphi 1 \to \pi 4\}$$
$$\textit{Ц}\,\{'\pi 1\,(1)\,''\varphi 1 \to \pi 2\}\,l$$
$$'('\varphi 1 \oplus ('\pi 4 \ominus 1) \bigotimes''\varphi 1 \oplus '\pi 2) \to r$$
$$\textit{Р}\,\{|\,'r\,| \leqslant |\,'\varphi 2\,|\,\} \downarrow \pi 4 \to \rho 1,\ '\pi 2 \to \delta 1,\ 'r \to '\varphi 2$$
$$l\ \ldots\ \textit{Р}\,\{'\pi 1 = '\rho 1\}\,l1$$
$$\textit{Ц}\,\{'\pi 1\,(1)\,''\varphi 1 \to \pi 4\}$$
$$'\varphi 1 \oplus ('\pi 1 \ominus 1) \bigotimes''\varphi 1 \oplus '\pi 4 \leftrightarrow '\varphi 1 \oplus ('\varphi 1 \ominus 1)\,''\varphi 1 \oplus '\pi 4$$
$$l1\ \ldots\ \textit{Р}\,\{'\pi 1 = '\delta 1\}\,l2$$
$$\textit{Ц}\,\{'\pi 1\,(1)\,''\varphi 1 \to \pi 4\}$$
$$'\varphi 1 \oplus ('\pi 4 \ominus 1)\,''\varphi 1 \oplus '\pi 1 \leftrightarrow '\varphi \oplus ('\pi 4 \ominus 1) \oplus '\delta$$
$$l2\ \ldots\ \textit{Я}.$$

(Строками с метками l и l_1 обеспечивается проверка необходимости перестановки строк и столбцов).

Подпрограмма *нули*

$$\textit{нули}\ \ldots\ \varnothing \to \varphi 1,\ \varnothing \to \pi 1$$
$$'\varphi 1 \oplus ('\pi 1 \ominus 1) \bigotimes^2 \varphi 1 \to l$$
$$\textit{Ц}\,\{'\pi 1 \oplus 1\,(1)^2\,\varphi 1 \to \pi 2\}$$
$$'('l \oplus '\pi 2) : '('l \oplus '\pi 1) \to 'l \oplus '\pi 2$$

$$Ц\,\{'\pi 1 \oplus 1\,(1)^2\,\varphi 1 \to \pi 2\}\,M$$
$$'('\varphi 1 \oplus ('\pi 2 \ominus 1) \times {}^2\varphi 1 \to l1$$
$$P\,\{'\,('l1 \oplus '\pi 1) = 0\}\,M1$$
$$Ц\,\{'\pi 1 \oplus 1\,(1)^2\,\varphi 1 \to \pi 3\}$$
$$'('l1 \oplus '\pi 3) : '('l1 \oplus '\pi 1) — '('l \oplus '\pi 3) \to 'l1 \oplus '\pi 3$$
$$M1\ \ldots\ M\ \ldots\ Я$$

3.* АДРЕСНАЯ РЕАЛИЗАЦИЯ НОРМАЛЬНЫХ АЛГОРИТМОВ МАРКОВА

В теории нормальных алгоритмов Маркова [12] информация о задаче задается как слово в некотором алфавите. Свойства слов определяются их записью. В качестве элементарной алгоритмической операции Марков ввел операцию замены определенных вхождений данного слова другими определенными словами.

Операции замены вхождений записываются в виде соответствий

$$A \to B,$$

называемых формулами подстановки, где A и B слова в заданном алфавите, стрелка — буква, не принадлежащая алфавиту. Применение операции замены как элементарной операции алгоритмического процесса к слову M означает следующее.

Если в слове M имеются вхождения слова A, то первое из этих вхождений заменяется на слово B. Например, пусть в алфавите $\{\alpha, \beta, \gamma, \delta\}$ задана подстановка

$$\beta\gamma \to \alpha\beta$$

и слово $\alpha\beta\gamma\gamma\beta\gamma\delta$. В результате замены первого вхождения $\beta\gamma$ на $\alpha\beta$ получаем слово $\alpha\alpha\beta\gamma\beta\gamma\delta$. Подстановка

$$A \to B$$

называется не применимой к слову M, если не существует ни одного вхождения слова A в слово M.

Нормальный алгоритм Маркова выражается списком формул подстановок в заданном алфавите, называемым схемой данного алгоритма.

Алгоритмический процесс согласно схеме алгоритма применительно к данному слову P состоит в следующем. В списке формул подстановок ищется первая подстановка, левая часть которой входит в слово P. Первое вхожде-

ние этой левой части подстановки в P заменяется правой частью подстановки; в результате получается слово P_1. Далее с P_1 делается то же самое, что и с P, и т. д.

Некоторые формулы подстановок в списке называются заключительными, в них после стрелки ставится точка:

$$A \to \cdot B$$

(точка — буква, не принадлежащая исходному алфавиту).

Остальные формулы называются простыми. Алгоритмический процесс прекращается либо из-за неприменимости ни одной из формул, либо после применения одной из заключительных формул. Результатом преобразования исходного слова является слово, полученное после прекращения алгоритмического процесса.

Пример. *Нормальный алгоритм Маркова для нахождения модуля разности двух целых положительных чисел.* Для записи целых положительных чисел используем рассмотренный ранее алфавит из одной буквы I, в качестве разделительного знака примем знак $*$. Тогда информация о задаче, состоящей в определении модуля разности двух чисел 5 и 7, запишется в виде слова

$$\text{IIII} * \text{IIIIII}.$$

Алгоритм может быть записан в виде списка из двух формул подстановок

$$\text{I} * \text{I} \to *$$
$$* \to$$

(вторая формула означает, что $*$ заменяется пустым знаком; заключительных формул в данном алгоритме нет).

Последовательные преобразования исходного слова IIII $*$ IIIIII будут иметь вид

IIII $*$ IIIIII
III $*$ IIII
II $*$ III
I $*$ II
$*$ I
I

Рассмотрим пример с одной заключительной формулой подстановки — нормальный алгоритм определения целой части от деления целого числа на число 3 в алфавите $\{1, *, a\}$:

$$* a \to \text{I} *$$
$$* \to \cdot$$
$$\text{III} \to a$$
$$\text{I} \to$$
$$a \to * a$$

Знак I используется, как и прежде, для записи целых чисел. Буквы $*$ и α используются в процессе работы алгоритма.

Рассмотрим несколько более сложный алгоритм деления целых чисел, используя алфавит {I, $\times$, α, β, $*$, $\square$, $\boxtimes$}.

Буква I используется, как и в предыдущих примерах, для записи целых чисел. Исходная информация задается в виде слова, состоящего из двух целых чисел, разделенных буквой $*$; слева от знака $*$ записывается делимое, справа — делитель. Остальные буквы используются в процессе работы алгоритма.

Результатом является слово, состоящее из двух целых чисел — остатка (слева) и частного (справа), записанных в том же алфавите {I} и разделенных буквой $*$. Если частное равно нулю, то результирующим словом будет остаток; если остаток равен нулю, то результирующим словом будет частное, перед которым стоит знак $*$; если и остаток и частное равны нулю (делимое равно нулю), результирующим словом будет пустое слово. Алгоритм неприменим при делении на нуль.

Нормальный алгоритм деления целых чисел

$$
\begin{aligned}
&\mathrm{I} * \mathrm{I} \to * \beta\\
&\beta\,\mathrm{I} \to \mathrm{I}\,\beta\\
&* \beta \to \times\, \alpha\beta\\
&\alpha\beta \to \beta\alpha\\
&\beta\alpha \to \beta\,\square\,\alpha\\
&\alpha\,\square\,\alpha \to \alpha\alpha\\
&* \mathrm{I} \to \boxtimes\\
&\boxtimes\,\mathrm{I} \to \boxtimes\\
&\boxtimes \to\\
&\beta \to \mathrm{I}\\
&\times \to *\\
&\alpha \to \mathrm{I}\\
&\square \to *.
\end{aligned}
$$

Из приведенных примеров видно сходство схем нормальных алгоритмов и программ для электронных вычислительных машин.

По известному тезису Маркова произвольный алгоритм может быть представлен в виде нормального алгоритма. Иначе говоря, не существует такого вычисления, которое нельзя было бы осуществить с помощью некоторого алгоритма Маркова.

Тезис Маркова базируется на том, что, во-первых, алгоритмы Маркова равносильны всем известным алгоритмам, и, во-вторых, с помощью алгоритмов Маркова может быть осуществлено произвольное алгоритмическое преобразование информации, практически уже осуществлявшееся. Однако уже для несложных вычислений, например выполнения арифметических операций, приходится строить весьма громоздкие алгоритмы Маркова.

Приведем универсальную адресную программу реализации нормальных алгоритмов Маркова, являющуюся еще одним доказательством полноты адресного языка. Из описания работы алгоритма Маркова видно, что, прежде чем приступить к его реализации, целесообразно составить подпрограмму распознавания первого вхождения слова A в слово X и подпрограмму выполнения подстановки.

1. Подпрограмма распознавания первого вхождения слова A в слово X. Предположим, что информация о словах X и A задается в виде α-последовательностей с ведущими адресами ξ и η соответственно; ζ — адрес, по которому выдается выходная информация. Если $A \subset X$, то по адресу ζ подпрограмма засылает нуль, в противном случае по адресу ζ выдается номер первой буквы вхождения. Формула вхождения подпрограммы

$$\boldsymbol{П}\ Марков\ 1\{\xi,\ \eta,\ \zeta\},$$

а сама подпрограмма запишется в виде

$$Марков\ 1\ \ldots\ \varnothing \Rightarrow \psi 1,\ \varnothing \Rightarrow \psi 2,\ \varnothing \Rightarrow \varphi 1$$
$$0 \Rightarrow' \varphi 1$$
$$\boldsymbol{Ц}\{0\,(1)^2\,\psi 1\ \ominus^2 \psi 2 \Rightarrow \pi\}\ M4$$
$$\boldsymbol{Ц}\{1\,(1)^2\,\psi 2 \Rightarrow l\}\ M3$$
$$\boldsymbol{Р}\{'('\psi 1 \oplus' \pi \oplus' l) = '('\psi 2 \oplus' l)\} \downarrow M2$$
$$M3\ \ldots\ '\pi \oplus 1 \Rightarrow '\varphi 1$$
$$M2\ \ldots\ M4\ \ldots\ Я.$$

В программе последовательно сравниваются буквы слова A с буквами слова X, начиная с первых; в случае несовпадения всех букв A с отрезком слова X сравнение повторяется с буквы слова X, следующей за той, с которой начиналось сравнение в предыдущий раз, до тех пор, пока не будет найдено вхождение A в X, либо пока остаток слова X не станет короче слова A.

2. Подпрограмма выполнения подстановки. Информацией для нее служат: слова X и B, записанные в виде $\alpha 1$- и $\alpha 2$-последовательностей; адреса, хранящие номер первой буквы вхождения слова A в слово $X(\zeta 1)$ и номер первой буквы после вхождения A $(\zeta 2)$. Новое слово $X1$ получается в той же $\alpha 1$-последовательности. Формула вхождения подпрограммы

$$\boldsymbol{П}\ Марков\ 2\ \{\alpha 2,'\ \zeta 1,\ '\zeta 2,\ \alpha 1\}.$$

Подпрограмма может быть записана в виде

$$\begin{gathered}
\textit{Марков } 2 \ldots \varnothing \Rightarrow \psi 1,\ \varnothing \Rightarrow \psi 2,\ \varnothing \Rightarrow \psi 3,\ \varnothing \Rightarrow \varphi 1 \\
{}^{2}\varphi 1 \ominus {}'\psi 3 \Rightarrow r;\ {}'\psi 3 \ominus {}'\psi 2 \ominus {}'\psi 1 \Rightarrow \Delta,\ {}^{2}\psi 1 \ominus {}'\Delta \Rightarrow {}'\varphi 1 \\
\boldsymbol{P}\{{}'\Delta = 0\}\, M1 \\
{}'\Delta : |\,{}'\Delta\,| \Rightarrow \delta \\
\boldsymbol{Ц}\{(1 \ominus {}'\delta) \otimes \tfrac{{}'r}{2}\,({}'\delta)\,(1 \oplus {}'\delta) \otimes \tfrac{{}'r}{2} \Rightarrow \pi\} \\
{}'({}'\varphi 1 \oplus {}'\psi 3 \oplus {}'\pi) \Rightarrow {}'\varphi 1 \oplus {}'\psi 3 \oplus {}'\pi \ominus {}'\Delta \\
M1 \ldots \boldsymbol{Ц}\{1\,(1)^2\,\psi 1 \Rightarrow \pi\},\ \textit{Я} \\
{}'({}'\psi 1 \oplus {}'\pi) \Rightarrow {}'\varphi 1 \oplus {}'\psi 2 \oplus {}'\pi \ominus 1.
\end{gathered}$$

Во второй строке вычисляется длина части слова X, стоящей после вхождения A, разность длин слов X и $X1$, длина слова $X1$. Если длины слов A и B равны (${}'\Delta = 0$), то B просто пересылается на место A (цикл с меткой $M1$).

По адресу δ помещается шаг параметра цикла; если слово A длиннее B (${}'\Delta > 0$), остаток X сдвигается влево, а если слово A короче слова B (${}'\Delta < 0$) — вправо.

Составим основной алгоритм. Исходной информацией для него служит слово X, состоящее из упорядоченной последовательности букв, записанных в виде α-последовательности с фиксатором $\psi 4$, и формул подстановок $A_1 \to B_1,\ A_2 \to B_2,\ \ldots,\ A_n \to B_n$, которые располагаются в виде β-последовательности так, что ${}'\beta = n_1$ — число букв в слове A_1, ${}'(\beta \oplus 1)$ — первая буква слова A_1 и т. д.; ${}'(\beta \oplus {}'\beta \oplus 1) = m_1$ — число букв в слове B_1, ${}'(\beta \oplus {}'\beta \oplus 2)$ — первая буква слова B_1 и т. д.; ${}'(\beta \oplus n_1 \oplus m_1 \oplus 2) = {}'(\beta \oplus {}'\beta \oplus 2 \oplus {}'(\beta \oplus {}'\beta \oplus 1)) = з$ или $нз$ — признак того, что подстановка $A_1 \to B_1$ — заключительная ($з$) или нет ($нз$); ${}'(\beta \oplus n_1 \oplus m_1 \oplus 3)$ — число букв в слове A_2 и т. д. Последний адрес последовательности хранит признак конца $я$. Фиксатор этой последовательности $\psi 5$.

Программа запишется в виде

$$\begin{gathered}
\textit{Марков} \ldots {}'\psi 5 \Rightarrow \eta \\
M\ 1 \ldots \boldsymbol{П}\ \textit{Марков}\ 1\{{}'\psi 4,\ {}'\eta,\ \zeta\} \\
{}'\eta \oplus {}^{2}\eta \oplus 1 \Rightarrow r \\
\boldsymbol{P}\{{}'\zeta = 0\}\, M2 \\
\boldsymbol{П}\ \textit{Марков}\ 2\{{}'r,\ {}'\zeta,\ {}'\zeta \oplus {}^{2}\eta,\ {}'\psi 4\} \\
\boldsymbol{P}\{{}'({}'r \oplus {}^{2}r \oplus 1) = з\}!\downarrow \textit{Марков} \\
M2 \ldots {}'r \oplus {}^{2}r \oplus 1 \Rightarrow \eta \\
\boldsymbol{P}\{{}'({}'\eta \oplus 1) \neq я\}\, M1 \downarrow !.
\end{gathered}$$

По первой строке просматриваются формулы подстановок, начиная с первой. Вторая строка означает переход к выполнению подпрограммы поиска вхождения левой части подстановки в слово X. Если вхождения нет, рассматриваем следующую формулу подстановки, в противном случае переходим к подпрограмме выполнения операции подстановки, после которой для заключительных подстановок согласно шестой строке алгоритм заканчивает работу. По седьмой строке осуществляется засылка в η первого адреса последовательности букв следующей подстановки.

ГЛАВА V

АДРЕСНЫЙ ЯЗЫК И МЕТОД УНИВЕРСАЛЬНЫХ ПРОГРАММИРУЮЩИХ ПРОГРАММ

1. ПОСТАНОВКА ЗАДАЧИ

Задача программирующей программы — перевод исходной информации записи алгоритма в некотором абстрактном алгоритмическом языке в программу для конкретной машины — машинный код. Среди известных алгоритмических языков адресный язык в этом отношении имеет известные преимущества. Перевод из общего адресного языка на язык конкретной машины с помощью программирующей программы может быть разделен на этапы введением ряда стилей, все более и более приближающихся к языку машины. Таким образом, работа программирующей программы может быть сведена, в основном, к формальным преобразованиям внутри адресного языка — к переводу из стиля в стиль [22].

Граница, до которой необходимо вручную доводить программы задач с тем, чтобы дальнейшую их обработку проводила программирующая программа, выбирается произвольно в зависимости от того, насколько сложную программирующую программу целесообразно осуществить в данный момент. Таким образом, можно практически автоматизировать программирование на определенном уровне.

Создание сложных программирующих программ, переводящих запись алгоритма с самого абстрактного алгоритмического языка на язык конкретной машины, не единственный и, возможно, не лучший путь автоматизации программирования. Он имеет, на наш взгляд, ряд недостатков:

1. Для разработки таких программ и их проверки и отладки необходимы длительные сроки, поэтому вопрос о текущей автоматизации ставится по принципу «все или ничего».

2. Такие программы неизбежно будут невероятно длинными. Следовательно, они будут работать с многократным обращением к внешней памяти: большую часть времени машина будет пересылать информацию из памяти в память.

3. Ввиду большого расстояния между входным и выходным языками трудно добиться оптимального программирования для всех задач.

4. Как и во всех больших программах, имеется большая вероятность ошибок, которые трудно локализовать.

Поэтому представляется рациональным другой путь автоматизации программирования. Прежде всего устанавливаются несколько стандартов адресного языка, каждый из которых характеризуется распределением памяти (уровнем) и ограничением выразительных средств (стилем). Одним из стандартов является свободный язык. Кроме того, имеется ряд машинных языков.

Предполагается существование программ перевода из некоторых стандартов языка в другие или в машинный язык. Каждый такой перевод приближает запись алгоритма к машинному языку. Стандартов должно быть как можно меньше, но достаточно для того, чтобы программы перевода помещались в оперативной памяти машин вместе с большим куском исходной информации. Переработка записи алгоритма в программу машин сводится к обработке его несколькими программами перевода.

Преимущества этого способа таковы:

1. Автоматизация программирования производится в короткие сроки и дает значительные результаты уже на первых порах, что проверено на опыте ряда ПП.

2. Программы перевода, полностью находящиеся в оперативной памяти, работают быстро.

3. Ввиду меньшего размера ПП ошибки в них менее вероятны.

4. Такая система автоматизации способна к развитию и усовершенствованию. Любую программу перевода можно изменять, не нарушая работы всей системы; каждое такое улучшение немедленно дает свои результаты.

5. Предложенная система не исключает первого способа автоматизации. Достаточно склеить несколько программ перевода, чтобы получить программирующую программу первого типа.

Особенно важно отметить, что благодаря программирующим программам расширяется круг лиц, способных

подготовлять программы для машин в адресном виде, из числа специалистов, занимающихся непосредственной алгоритмизацией тех или иных процессов переработки информации и совсем незнакомых с кодами конкретных машин и не знающих наперед, какая из машин будет исполнителем программы. Последнее заслуживает особого внимания в связи с возрастающей потребностью в специалистах, вызванной увеличением числа машин и огромным повышением их быстродействия. Работа, обычно относимая к специальности программиста, теперь может быть разделена на две существенно разные части. Уточнение и составление алгоритмов выполняется человеком. Результатом его работы является адресная программа. Перевод последней в машинный код осуществляется программирующими программами. Труд по обслуживанию программирующих программ можно свести до минимума внедрением читающих автоматов.

2. О ПРИНЦИПАХ ПОСТРОЕНИЯ ПРОГРАММИРУЮЩИХ ПРОГРАММ

Программирующая программа является, по существу, переводящей программой. Однако алгоритмы перевода ПП ввиду особой структуры алгоритмических языков являются более простыми, более определенными и исчерпывающими, чем алгоритмы перевода, например, с английского языка на русский.

Работа ПП основывается на заранее установленных основных соответствиях между определенными синтаксическими выражениями алгоритмического языка, которые назовем представимыми, и кодами машины. Количество и характер представимых выражений должны быть достаточными для расчленения любого алгоритма на цепочку таких выражений. ПП должна в ходе своей работы представлять исходную информацию в виде такой цепочки и, используя основные соответствия, заменять эти выражения одно за другим кодами машины. Выбор системы представимых выражений (т. е. установление основных соответствий) определяет структуру и принципы построения ПП.

Большинство созданных до сих пор ПП основаны на одном и том же принципе. В них множество представимых выражений составляется из синтаксических выраже-

ний, переводимых на машинный язык одной или несколькими командами. Работа ПП, основанная на этом принципе, сводится к следующему:

1. ПП, перебирая исходную информацию, отыскивает в ней представимое выражение (требуется, чтобы в любой исходной информации всегда имелись такие выражения).

2. В формируемую программу записывается приказ (группа приказов), соответствующий найденному выражению.

3. Соответственно преобразуется исходная информация, и цикл работы ПП начинается опять с п. 1.

4. Требуется, чтобы в результате преобразования информации в ней до ее полного исчерпания появлялись представимые выражения. Предполагается, что выполнение представимых выражений в преобразуемом алгоритме в порядке их нахождения ПП — одно из допустимых в алгоритме порядков действий.

На этом принципе построены программирующие программы, описанные в [3], [20] и др.

Выполнение перечисленных требований достигается за счет достаточной сложности ПП или дополнительных требований к записи алгоритма (например, бесскобочная запись формул по Лукасевичу, которая принята для ПП-АК [1]). Так при скобочной записи арифметических формул представимой совокупностью будет, например, выполнимая операция — знак двухместной операции, соединяющий два соседних с ним символа величин, или знак одноместной операции, за которым непосредственно следует величина. В отведенном рабочем массиве программирующая программа по найденному представимому сочетанию кодов записывает одну или несколько команд (в зависимости от адресности машины), выбирая при необходимости рабочую ячейку для результата, и заменяет в исходной информации это сочетание символов номером рабочей ячейки. Далее в той или иной форме осуществляется сжатие исходной информации, поскольку количество ее элементов уменьшается при такой обработке, и процесс повторяется вплоть до исчерпания всей записи. Реализация тех или иных усовершенствований (например, экономии рабочих ячеек) не вызывает принципиальных затруднений.

Исходя из специфики алгоритмических языков, в которых каждый из синтаксических знаков (или их неко-

торая последовательность) в отличие от живых языков несет свою собственную информацию о смысле, не зависимую от сочетаний с другими знаками, можно предложить следующий принцип поэлементного перевода адресных формул в машинный код.

Будем считать представимыми выражениями минимальные выражения, составленные из основных символов принятого в языке алфавита и несущие синтаксическую информацию. Если, например, рассматриваются простейшие арифметические формулы в скобочной записи, то представимым будет каждое из следующих выражений: величина, адрес величины, $+$, $-$, $\times$, $:$, (,), sin и др.

Принятое множество представимых выражений определяет основные соответствия. Прежде всего, представимые выражения так малы, что им соответствуют не команды машины, а обычно, часть команды — отдельный код. Кроме того, в число представимых элементов входят, например, скобки, определяющие не действия, а порядок их выполнения. Им должно быть поставлено в соответствие, кроме, может быть, кодов, еще и установление какого-то порядка записи команд в формируемой программе.

Работу ПП, основанной на этом принципе, можно характеризовать следующим:

1. Исходная информация просматривается однократно, в естественном порядке, поэлементно.

2. Для формирования программы отдельной формулы исходного алгоритма выделено рабочее поле. В это поле вносятся те или иные записи в зависимости от просматриваемого в данный момент элемента. Отдельным элементам может соответствовать не только внесение кодов в рабочее поле, но и определение мест в поле для внесения следующих кодов.

3. Ни одна часть формируемой программы не может считаться вообще говоря законченной, пока не запрограммирована вся формула. Поформульная обработка информации вызвана тем, что формула, во всяком случае, имеет законченный смысл.

Рассмотрим в качестве примера алгоритм перевода адресных функций в код трехадресной машины, основанный на принципе поэлементной расшифровки.

Для простоты предположим, что в запись адресной функции могут входить: адреса произвольного ранга, за исключением нулевого; произвольные двухместные

операции $O2$ арифметические или логические, которым соответствует трехадресная команда вида

$O2$	a	b	c

$('aO2'b \Rightarrow c)$,

а также одноместные операции $O1$, которым соответствуют следующие машинные команды:

$O1$	a	b	c

.

Здесь в нулевом адресе указывается $O1$ — вид операции (начальный адрес соответствующей подпрограммы), в первом адресе a — адрес аргумента, во втором адресе b — адрес следующей команды машинной программы (ячейка возврата с подпрограммы) и в третьем адресе c — адрес рабочей ячейки, по которому засылается результат операции — значение соответствующей функции, выдаваемое подпрограммой.

Кроме этого, предположим, что понижение ранга адреса осуществляется специальной командой вида

Ф	a		b

(см., например, машину «Киев» [1]), действие которой соответствует засылке

$$''a \Rightarrow b.$$

Примем строго скобочную запись формул, т. е. запись, в которой порядок выполнения операций определяется только скобками (старшинство операций не учитывается), лишних скобок нет. Описание строго скобочной записи адресных функций дадим с помощью металингвистических формул [15]. Последовательности знаков, заключенных в скобки $\langle\ \rangle$, представляют собой металингвистические переменные, значениями которых являются последовательности символов; знаки $::=$ и $|$ являются металингвистическими связками (знак $::=$ означает «равно по определению», знак $|$ означает «или»). Знак в формуле, не являющийся переменной или связкой, обозначает самого себя или подобный ему класс знаков. Соединение знаков или переменных означает соединение обозначаемых ими последовательностей.

Согласно принятому условию основными символами являются:

(— открывающая скобка;

) — закрывающая скобка;

$O1$ — одноместные операции, к которым отнесем также штрих-операцию;

$O2$ — двухместные операции;

a — адреса.

Таким образом,

⟨основной символ⟩ : : = (|) | $O1$ | $O2$ | a

⟨компонента⟩ : : = a | $O1$ ⟨компонента⟩ | (⟨компонента⟩ $O2$ ⟨компонента⟩)

⟨строго скобочная запись функции⟩ : : = a | $O1$ ⟨компонента⟩ | ⟨компонента⟩ $O2$ ⟨компонента⟩.

Формула дает рекурсивное правило для образования строго скобочной записи адресных функций.

Рабочий массив алгоритма ПАФ, отведенный для составления программы, заполняется снизу вверх. Компоненты команд устанавливаются по мере просмотра закодированной информации (в процессе работы некоторые команды могут сформироваться полностью, в то время как другие, быть может, и те, которые будут выполняться раньше, еще вовсе не сформированы или сформированы частично). Попутно производится экономия рабочих ячеек.

Для описания алгоритма ПАФ введем обозначения:

C — операция следования в последовательности адресов элементов исходной информации; первый адрес обозначен через α (в одном машинном адресе может находиться несколько элементов);

φ — фиксатор элементов входной информации; вначале $'\varphi = C^{-1}\alpha$;

ψ — фиксатор рабочего массива, отведенного для записи команд рабочей программы; адрес последней ячейки этого массива l;

$\varkappa$ — фиксатор массива рабочих ячеек рабочей программы (РП); r — первая ячейка этого массива;

Σ, λ — рабочие фиксаторы алгоритма ПАФ.

Фиксатор Σ используется при обработке скобок. С помощью этого фиксатора в некотором массиве рабочих ячеек алгоритма ПАФ из k адресов ($\gamma \oplus 1$, $\gamma \oplus 2$, ..., $\gamma \oplus k$) при обработке символов открывающих скобок фиксируются необходимые адреса машинных команд,

к формированию которых совершается переход при обработке соответствующего (парного) символа закрывающей скобки (размер массива k определяет допустимую глубину скобок в выражениях адресных функций, которые данный алгоритм ПАФ может переводить в машинные программы).

Фиксатор λ используется для запоминания адреса машинной команды, к составлению которой необходимо перейти после обработки знаков штрих рангов выше первого или знака одноместной операции.

Условимся считать, что *пр*f означает признак элемента f. Среди признаков различаем:

ω — признак конца записи;
'адр — признак адреса первого ранга;
(— открывающая скобка;
) — закрывающая скобка;
'не-адр — признак наличия штрих-операции, не относящейся непосредственно к адресу;
*О*1 — признак одноместной операции;
*О*2 — признак двухместной операции;
раб — признак рабочей ячейки.

В алгоритме признаки элементов ω, *'адр*, (,), *'не-адр*, *О*1, *О*2 используются в качестве меток.

Через $'\psi_0$, $'\psi_I$, $'\psi_{II}$, $'\psi_{III}$ обозначены соответственно содержимые 0-го, I, II, III адресов ячейки, фиксируемой ψ. Записи машинных кодов, используемых в адресной программе, взяты в прямоугольные рамки.

Фиксатор φ обозревает элементы входной информации (адресного алгоритма), т. е. в каждый данный момент работы алгоритма $^2\varphi$ является обрабатываемым элементом. В зависимости от признака элемента по формуле $пр^2\varphi$ совершается переход к строке, помеченной соответствующей меткой. Выражение $пр^2\varphi$ может принимать одно из семи значений: ω, *'адр*, *'не-адр*, (,), *О*1, *О*2; таким образом, эта формула означает переход на одну из семи ветвей алгоритма, каждая из которых помечена соответствующей меткой.

Алгоритм перевода адресных функций для условной III-адресной машины

$$ПАФ \ \ldots \ 0 \Rightarrow \lambda;\ r \Rightarrow \varkappa;\ l \Rightarrow \psi;\ \gamma \oplus 1 \Rightarrow \Sigma$$
$$H \ \ldots \ C('\varphi) \Rightarrow \varphi$$
$$пр^2\varphi$$

$$(\ldots\ \boldsymbol{P}\{'\lambda = 0\}\,'\psi \Rightarrow {}'\Sigma \downarrow {}'\lambda \Rightarrow {}'\Sigma;\ 0 \Rightarrow \lambda$$
$$'\Sigma \bigoplus 1 \Rightarrow \Sigma$$
$$\boldsymbol{P}\{'('\psi_I) \neq 0\}\,(b$$
$$'\chi \Rightarrow {}'\psi_I;\ '\psi \bigominus 1 \Rightarrow \psi$$
$$(a\ \ldots\ '\chi \Rightarrow {}'\psi_{III};\ H$$
$$(b\ \ldots\ \boldsymbol{P}\{np'\,('\psi_I) = раб\}\,'\chi \bigoplus 1 \Rightarrow \chi$$
$$'\chi \Rightarrow {}'\psi_{II}$$
$$\boldsymbol{Ц}\{'\psi\,(\bigominus 1)\,\boldsymbol{P}\{{}^2\psi \neq 0\} \Rightarrow \psi\}$$
$$L\ \ldots\ (a$$
$$)\ \ldots\ '\Sigma \bigominus 1 \Rightarrow \Sigma;\ 0 \Rightarrow \lambda;\ {}^2\Sigma \Rightarrow \psi;\ H$$
$$O2\ \ldots\ \boldsymbol{P}\{'\lambda \neq 0\}\,'\lambda \Rightarrow \psi,\ 0 \Rightarrow \lambda$$
$${}^2\varphi \Rightarrow {}'\psi_0;\ H$$
$$Штрих\ a\ \ldots\ \boldsymbol{P}\{'('\psi_I) = 0\}^2\varphi \Rightarrow {}'\psi_I \downarrow {}^2\varphi \Rightarrow {}'\psi_{II}$$
$$H$$
$$Штрих\ \ldots\ \boldsymbol{P}\{'('\psi_I) \neq 0\}\,Шб$$
$$'\chi \Rightarrow {}'\psi_I$$
$$\boldsymbol{P}\{'\lambda = 0\}\,'\psi \Rightarrow \lambda$$
$$'\psi \bigominus 1 \Rightarrow \psi;\ Шв$$
$$Шб\ \ldots\ \boldsymbol{P}\{np'\,('\psi_I) = раб\}\,'\overset{1}{\chi} + 1 \Rightarrow \chi$$
$$'\chi \Rightarrow {}'\psi_{II}$$
$$\boldsymbol{Ц}\{'\psi\,(\bigominus 1)\,\boldsymbol{P}\{{}^2\psi \neq 0\} \Rightarrow \psi$$
$$L1\ \ldots$$
$$Шв\ \ldots\ \boxed{\Phi \mid \mid \mid {}'\chi} \Rightarrow {}'\psi$$
$$H$$
$$O1\ \ldots\ \boldsymbol{P}\{{}^2\psi_I \neq 0\}\,a$$
$$'\chi \Rightarrow {}'\psi_I$$
$$\boldsymbol{P}\{'\lambda = 0\}\,'\psi \Rightarrow \lambda$$
$$'\psi \bigominus 1 \Rightarrow \psi$$
$$в\ \ldots\ \boxed{{}^2\varphi \mid \mid {}'\psi \bigoplus 1 \mid {}'\chi} \Rightarrow {}'\psi;\ H$$
$$a\ \ldots\ \boldsymbol{P}\{{}^2\psi_I = раб\}\,'\chi + 1 \Rightarrow \chi$$
$$'\chi \Rightarrow {}'\psi_{II}$$
$$\boldsymbol{Ц}\{'\psi\,(\bigominus 1)\,\boldsymbol{P}\{{}^2\psi \neq 0\} \Rightarrow \psi\}\,L2$$
$$L2\ \ldots\ в$$
$$\omega\ \ldots\ Я$$

Проследим за работой этого алгоритма. Пусть дана адресная функция

$$(''a + \sin('b - \cos''d)) \times \operatorname{tg}\cos(''c + 'e)\,\omega,$$

где ω — признак конца записи.

Вначале

$$'\lambda = 0,\quad '\chi = r;\quad '\psi = l;\quad '\Sigma = \gamma \oplus 1;\quad '(l - i) = 0$$
$$\text{для } i = 0,\ 1,\ 2,\ \ldots$$

Будем отмечать изменения адресного отображения на каждом шаге работы алгоритма ПАФ.

1. Код открывающей скобки; $np^2\varphi = (:$

$$'(\gamma \oplus 1) = l$$
$$'\Sigma = \gamma \oplus 2$$

$'l =$

	r		

$$'\psi = l \ominus 1$$

$'(l \ominus 1) =$

			r

2. Код «$'$»; $np^2\varphi$-штрих;

$'(l \ominus 1) =$

	r		r

$$'\lambda = l \ominus 1$$
$$'\psi = l \ominus 2$$

$'(l \ominus 2) =$

Φ			r

.

3. Код $'a$; $np^2\varphi$-штрих a:

$'(l \ominus 2) =$

Φ	a		r

4. Код «+»; $np^2\varphi = O2$:

$$'\psi = l \ominus 1$$
$$'\lambda = 0$$

$'(l \ominus 1) =$

$+$	r		r

.

5. Код sin; $np^2\varphi = O1$:

$$'\chi = r \oplus 1$$

$$'(l \in 1) = \boxed{+ \mid r \mid r \oplus 1 \mid r}$$

$$\boxed{'\psi = l \ominus 3}$$

$$'(l \ominus 3) = \boxed{\sin \mid \quad \mid l \ominus 2 \mid r \oplus 1}$$

6. Код открывающей скобки; $np^2\varphi = (:$

$$'(\gamma \oplus 2) = l \ominus 3$$

$$'\Sigma = \gamma \oplus 3$$

$$'(l \ominus 3) = \boxed{\sin \mid r+1 \mid l \ominus 2 \mid r \oplus 1}$$

$$'\psi = l - 4$$

$$'(l \ominus 4) = \boxed{\quad \mid \quad \mid \quad \mid r \oplus 1}$$

7. Код $'b$; $np^2\varphi$-штрих a:

$$'(l \ominus 4) = \boxed{\quad \mid b \mid \quad \mid r \oplus 1}$$

8. Код «—»; $np^2\varphi = O2$:

$$'(l \ominus 4) = \boxed{- \mid b \mid \quad \mid r \oplus 1}$$

9. Код cos; $np^2\varphi = O1$:

$$'(l \ominus 4) = \boxed{- \mid b \mid r \oplus 1 \mid r \oplus 1}$$

$$'\psi = l \ominus 5$$

$$'(l \ominus 5) = \boxed{\cos \mid \quad \mid l \ominus 4 \mid r \oplus 1}$$

10. Код «′»; $np^2\varphi$-штрих:

$$'(l \ominus 5) = \boxed{\text{cos} \mid r \oplus 1 \mid l \ominus 4 \mid r \oplus 1}$$

$$'\lambda = l \ominus 5$$
$$'\psi = l \ominus 6$$

$$'(l \ominus 6) = \boxed{\Phi \mid \mid \mid r \oplus 1}$$

11. Код «′d»; $np^2\varphi$-штрих a

$$'(l \ominus 6) = \boxed{\Phi \mid d \mid \mid r \oplus 1}$$

12. Код закрывающей скобки; $np^2\varphi =)$:

$$'\Sigma = \gamma \oplus 2$$
$$'\lambda = 0$$
$$'\psi = l \ominus 3$$

13. Код закрывающей скобки; $np^2\varphi =)$:

$$'\Sigma = \gamma \oplus 1$$
$$'\psi = l$$

14. Код «×»; $np^2\varphi = O2$:

$$'l = \boxed{\times \mid r \mid \mid \mid}$$

15. Код tg; $np^2\varphi = O1$:

$$'\chi = r \oplus 2$$

$$'l = \boxed{\times \mid r \mid r \oplus 2 \mid}$$

$$'\psi = l \ominus 7$$

$$'(l \ominus 7) = \boxed{\text{tg} \mid \mid l \ominus 6 \mid r \oplus 2}$$

16. Код cos; $пр^2\varphi = O1$:

$$'(l \ominus 7) = \boxed{\text{tg} \mid r \oplus 2 \mid l \ominus 6 \mid r \oplus 2}$$

$$'\lambda = l \ominus 7$$
$$'\psi = l \ominus 8$$

$$'(l \ominus 8) = \boxed{\cos \mid \quad \mid l \ominus 7 \mid r \oplus 2}$$

17. Код открывающей скобки; $пр^2\varphi = ($

$$'(\gamma \oplus 1) = l \ominus 7$$
$$'\lambda = 0$$
$$'\Sigma = \gamma \oplus 2$$

$$'(l \ominus 8) = \boxed{\cos \mid r + 2 \mid l \ominus 7 \mid r \oplus 2}$$

$$'\psi = l \ominus 9$$

$$'(l \ominus 9) = \boxed{\quad \mid \quad \mid \quad \mid r \oplus 2}$$

18. Код «$'$»; $пр^2\varphi$-штрих:

$$'(l \ominus 9) = \boxed{\quad \mid r \oplus 2 \mid \quad \mid r \oplus 2}$$

$$\lambda = l \ominus 9$$
$$'\psi = l \ominus 10$$

$$'(l \ominus 10) = \boxed{\Phi \mid \quad \mid \quad \mid r \oplus 2}$$

19. Код «$'c$»; $пр^2\varphi$-штрих a:

$$'(l \ominus 10) = \boxed{\Phi \mid c \mid \quad \mid r \oplus 2}$$

20. Код «+»; $пр^2\varphi = O2$:

$$'\psi = l \ominus 9$$
$$'\lambda = 0$$

$$'(l \ominus 9) = \boxed{+ \mid r \oplus 2 \mid \quad \mid r \oplus 2}$$

21. Код «$'e$»; $пр^2\varphi$-штрих a:

$'(l \ominus 9) =$	$+$	$r \oplus 2$	e	$r \oplus 2$

22. Код закрывающей скобки; $пр^2\varphi =$):

$$'\Sigma = \gamma \oplus 1$$
$$'\psi = l \ominus 7$$

Вся программа для условной III-адресной машины имеет вид

$l \ominus 10$	Φ	c		$r \oplus 2$
$l \ominus 9$	$+$	$r \oplus 2$	l	$r \oplus 2$
$l \ominus 8$	cos	$r \oplus 2$	$l \ominus 7$	$r \oplus 2$
$l \ominus 7$	tg	$r \oplus 2$	$l \ominus 6$	$r \oplus 2$
$l \ominus 6$	Φ	d		$r \oplus 1$
$l \ominus 5$	cos	$r \oplus 1$	$l \ominus 4$	$r \oplus 1$
$l \ominus 4$	$-$	b	$r \oplus 1$	$r \oplus 1$
$l \ominus 3$	sin	$r \oplus 1$	$l \ominus 2$	$r \oplus 1$
$l \ominus 2$	Φ	a		r
$l \ominus 1$	$+$	r	$r \oplus 1$	r
l	$\times$	r	$r \oplus 2$	

О простоте алгоритмов перевода при последовательной обработке элементов исходной информации (алгоритмов, заданных в формальном языке) можно судить по приведенному примеру алгоритма ПАФ.

Благодаря приведенному принципу построения ПП можно по-новому подойти к решению вопроса автоматизации программирования. Несмотря на сложность ПП,

в процессе их работы далеко не полностью используются возможности машины по набору операций и в гибкости связей. Поэтому, учитывая еще частое применение ПП, желательно создать ПП в виде запаянной схемы, не зависимой от узлов машины. Последовательная обработка элементов исходной информации особенно удобна для читающих устройств. При таком способе автоматизации обеспечивается большая надежность за счет схемной реализации ПП, отсутствия кодировщиков и перфорации исходных данных на ленту.

На данном принципе построена приведенная в следующем параграфе этой главы программирующая программа для ЭВЦМ «Урал» ПП-Урал-ЭРА.

3. ОБ АЛГОРИТМИЗАЦИИ ПРОЦЕССА ПОСТРОЕНИЯ ПРОГРАММИРУЮЩИХ ПРОГРАММ

Построение программирующих программ сводится к установлению соотношений между представимыми выражениями входного языка и кодами конкретной машины. Поэтому структура ПП определяется, с одной стороны, выбором представимых выражений и, с другой,— особенностями языка ЭВЦМ. Построение ПП можно упростить, выбрав соответствующий стиль входного языка. Стиль, для которого запись алгоритма отличается от конкретной машинной записи только кодированием, назовем машинным стилем адресного языка.

Рассмотрим автоматизацию процесса построения ПП[1] путем разделения их работы на два этапа:

1) эквивалентные преобразования записи алгоритмов с выбранного стиля в машинный стиль;

2) кодирование.

Для простоты ограничимся рассмотрением алгоритмов программирования адресных функций, приняв для них строго скобочную запись, включающую следующие символы: двухместные операции *O*, открывающие и закрывающие скобки, операции засылки и штрих-адреса как единого символа (т. е. в записи не допускаются одноместные операции и допускаются лишь адреса первого ранга).

---

[1] Результаты этого параграфа получены под руководством автора аспирантом В. П. Семиком, см. До питання автоматизації побудови програмуючих програм, Обчислювальна математика і техніка, Зб. праць ІК АН УРСР, 1963, № 1.

Выделим три основные машинные стили: одно-, двух- и трехадресный. Свободному адресному языку соответствует машинный язык с нефиксированной структурой команд идеальной адресной машины (ИАВМ).

Каждой записи адресной функции с нефиксированной структурой соответствует определенная комбинация адресных формул избранного машинного стиля.

В одноадресном стиле такими формулами являются:

$$'A \Rightarrow S \quad (1)$$
$$'SO'B \Rightarrow S \quad (2)$$
$$'S \Rightarrow C \quad (3)$$

В двухадресном стиле:

$$'AO'B \Rightarrow S \quad (4)$$
$$'A \Rightarrow S \quad (5)$$
$$'SO'B \Rightarrow C \quad (6)$$
$$'S \Rightarrow C \quad (7)$$
$$'SO'B \Rightarrow S \quad (8)$$

В трехадресном стиле:

$$'AO'B \Rightarrow C \quad (9)$$
$$'A \Rightarrow C \quad (10)$$

Здесь A, B, C — произвольные адреса, S — фиксированный адрес-сумматор.

Трехадресный стиль отличается от двух- и одноадресного отсутствием сумматора.

Для фиксирования результата операции, записанной в скобках, перед которыми непосредственно не стоит знак двухместной операции, введем условный сумматор S и назовем получаемый при этом набор формул

$$'AO'B \Rightarrow C \quad (I)$$
$$'A \Rightarrow C \quad (II)$$
$$'AO'B \Rightarrow S \quad (III)$$
$$'A \Rightarrow S \quad (IV)$$
$$'SO'B \Rightarrow C \quad (V)$$
$$'S \Rightarrow C \quad (VI)$$
$$'SO'B \Rightarrow S \quad (VII)$$

обобщенным машинным стилем (с левым сумматором).

Формулы (III) и (VII) совпадут с формулами (9)—(10), если S — произвольный адрес. Формулам (I) и (II), отсутствующим в двухадресном стиле, соответствует последовательность формул (4)—(7) и (5)—(7). Между обобщенным машинным и одноадресным стилями можно установить следующее соответствие:

$$(I) \sim (1),\ (2),\ (3)$$
$$(II) \sim (1),\ (3)$$
$$(III) \sim (1),\ (2)$$
$$(V) \sim (2),\ (3)$$

Используя приведенные соответствия, легко перейти от обобщенного машинного стиля на конкретный машинный стиль. Этот перевод можно выполнить одновременно с кодированием. Тогда работа ПП для разных ЭВМ будет отличаться по существу лишь вторым этапом.

Для примера построим соответствующие алгоритмы кодирования для машин «Минск-I» и «Урал-I»[1]. Формулы обобщенного машинного стиля будем считать заданными в последовательности строк $\alpha \oplus 1$, $\alpha \oplus 2$, ..., каждая из которых разделена на 4 подадреса для размещения элементов формул. Например, формула I запишется в виде

$i \oplus k$	A	O	B	C

(символ $\rightarrow$ и штрихи в записи опущены). Для обозревания α-последовательности введем фиксатор φ:

$$'\varphi = i \oplus k,\ k = 1,\ 2,\ \ldots$$

и пусть $('\varphi)_i$ обозначает i-й подадрес элемента, содержащегося по адресу $'\varphi$. Формулам двухадресного стиля соответствуют следующие команды «Минска».

	1	2	3	I	II	
(4)		1		B	A	(11)
(5)		1			A	(11*)

[1] Описание машины «Минск-I» см., например, в книге И. Г. Илзиня, Программирование для двухадресных цифровых вычислительных машин, Изд. АН Латвийской ССР, Рига, 1962; описание машины «Урал-I» см. далее в гл. VI.

(6)		2		B	C	(12)
(7)		2			C	(12*)
(8)		3		B		(13)

Содержимое разрядов 1-го и 3-го определяется видом операции.

Команды в языке «Минска» разместим в j-последовательность, обозреваемую фиксатором ψ. Обозначим через $('\psi)_{\text{I}}$ и $('\psi)_{\text{II}}$ первый и второй адреса ячейки, обозреваемой фиксатором ψ, а через $('\psi)_{1.3}$ и $('\psi)_2$ — ее соответствующие разряды 1, 3 и 2-й.

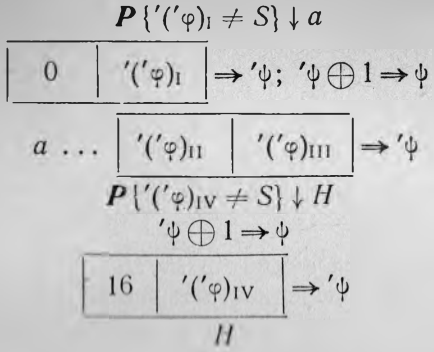
Соответствие между обобщенным стилем и языком «Минска» позволяет записать следующий алгоритм кодирования:

$$B \ \ldots \ i \Rightarrow \varphi, \ j \Rightarrow \psi$$
$$H \ \ldots \ '\varphi \oplus 1 \Rightarrow \varphi$$
$$\boldsymbol{P}\{{}^{2}\varphi \neq тчк\} \downarrow Я$$
$$'\psi \oplus 1 \Rightarrow \psi; \ M1 \Rightarrow \omega$$
$$\boldsymbol{P}\{'('\varphi)_{\text{I}} \neq S\} \downarrow 3 \Rightarrow ('\psi)_2; \quad M2 \Rightarrow \omega; \ M$$
$$1 \Rightarrow ('\psi)_2; \ '('\varphi)_{\text{I}} \Rightarrow ('\psi)_{\text{III}}$$
$$M \ \ldots \ '('\varphi)_{\text{II}} \Rightarrow ('\psi)_{1.3}; \ '('\varphi)_{\text{III}} \Rightarrow ('\psi)_{\text{I}}$$
$$\boldsymbol{P}\{'('\varphi)_{\text{IV}} \neq S\} \downarrow H$$
$$'\omega$$
$$M1 \ \ldots \ '\psi \oplus 1 \Rightarrow \psi$$
$$M2 \ \ldots \ 2 \Rightarrow ('\psi)_2$$
$$'('\varphi)_{\text{IV}} \Rightarrow ('\psi)_{\text{II}}$$
$$H$$

(через *тчк* обозначен признак конца записи).

Аналогично, устанавливая соответствие между формулами одноадресного стиля и командами «Урала-I», получаем кодирующий алгоритм

$$B \ \ldots \ i \Rightarrow \varphi, \ j \Rightarrow \psi$$
$$H \ \ldots \ '\varphi \oplus 1 \Rightarrow \varphi$$
$$\boldsymbol{P}\{{}^{2}\varphi \neq тчк\} \downarrow Я$$
$$'\psi \oplus 1 \Rightarrow \psi$$



Составив для заданных машин библиотеку алгоритмов кодирования, можно построить комплексную ПП. При построении конкретной ПП разделение во времени обоих этапов работы ПП может оказаться нерациональным, в связи с чем алгоритмы эквивалентных преобразований и кодирования могут объединяться. Используя соответствие между стилями, можно из объединенного алгоритма для одной ЭВМ строить алгоритмы для других машин.

Рассмотрим блок программирования адресных функций (БПАФ), основанный на поэлементной расшифровке информации для ЭВМ «Киев»[1]. Для простоты снова ограничимся алгоритмами обработки символов (,), *O*, *'A*. Приняв обозначения, введенные в примере предыдущего параграфа этой главы, и учитывая, что в нашем случае алгоритм обработки символов одноместных операций должен быть опущен, (надобность в фиксаторе λ соответственно отпадает), алгоритм ПАФ для «Киева*» перепишем в виде

$$B \ \ldots \ r = \chi;\ l \Rightarrow \psi;\ \gamma \bigoplus 1 \Rightarrow \Sigma$$
$$H \ \ldots \ C('\varphi) \Rightarrow \varphi$$
$$(\ldots \ \boldsymbol{P}\{'('\psi_{\mathrm{I}}) \neq 0\}\, b$$
$$'\chi \Rightarrow '\psi_{\mathrm{I}}$$
$$a \ \ldots \ '\psi \Rightarrow '\Sigma;\ '\Sigma \bigoplus 1 \Rightarrow \Sigma$$
$$Ц\{'\psi\,(\ominus 1)\,\boldsymbol{P}\{^{2}\psi \neq 0\} \Rightarrow \psi\}\,Б$$
$$Б \ \ldots \ '\chi \Rightarrow '\chi_{\mathrm{III}};\ H$$

[1] Набор операций машины «Киев» и более полное описание этого блока см. в работе [1].

$$b \ \ldots \ \boldsymbol{P}\{'('\psi_{\text{I}}) = \textit{раб}\}\,'\chi \oplus 1 \Rightarrow \chi$$
$$'\chi \Rightarrow '\psi_{\text{II}};\ a$$
$$O \ \ldots \ {}^{2}\varphi \Rightarrow '\psi_{0};\ H$$
$$\textit{Штрих}\ a \ \ldots \ \boldsymbol{P}\{'('\psi_{\text{I}}) = 0\}\,{}^{2}\varphi \Rightarrow '\psi_{\text{I}} \downarrow {}^{2}\varphi \Rightarrow '\psi_{\text{II}}$$
$$H$$
$$) \ \ldots \ '\Sigma \ominus 1 \Rightarrow \Sigma;\ {}^{2}\Sigma \Rightarrow \psi;\ H$$

Введем условный сумматор S. При этом приведенный блок можно будет переписать в виде

$$B \ \ldots \ k \Rightarrow \psi,\ r \Rightarrow \chi,\ \gamma \oplus 1 \Rightarrow \Sigma$$
$$H \ \ldots \ Ц\{'\varphi_{0},\ C \varnothing \Rightarrow \varphi\}$$
$$np^{2}\varphi$$
$$(\ldots \ '\psi \Rightarrow '\Sigma;\ '\Sigma \oplus 1 \Rightarrow \Sigma$$
$$P\{'('\psi)_{\text{I}} \neq 0\} \downarrow a$$
$$P\{'('\psi)_{\text{I}} = S\}\,'\chi \oplus 1 \Rightarrow \chi$$
$$'\chi \Rightarrow ('\psi)_{\text{II}}$$
$$Ц\{'\psi\,(\ominus 1)\,P\{{}^{2}\psi \neq 0\} \Rightarrow \psi\}$$
$$L \ \ldots$$
$$'\chi \Rightarrow ('\psi)_{\text{III}}$$
$$H$$
$$a \ \ldots \ 'S \Rightarrow ('\psi)_{\text{I}};\ '\psi \ominus 1 \Rightarrow \psi;\ 'S \Rightarrow ('\psi)_{\text{III}}$$
$$H$$
$$O \ \ldots \ {}^{2}\varphi \Rightarrow ('\psi)_{0}$$
$$H$$
$$A \ \ldots \ P\{'('\psi)_{\text{I}} = 0\}\ {}^{2}\varphi \Rightarrow ('\psi)_{\text{I}} \downarrow {}^{2}\varphi \Rightarrow ('\psi)_{\text{II}}$$
$$H$$
$$) \ \ldots \ '\Sigma \ominus 1 \Rightarrow \Sigma$$
$$''\Sigma \Rightarrow \psi$$
$$H.$$

Приведенный алгоритм выдает только команды вида

O	A	B	C	(14)

O	A	B	S	(15)

O	*S*	*B*	*C*	(16)
O	*S*	*B*	*S*	(17)

Командам «Киева» соответствуют следующие формулы обобщенного машинного стиля:

$$\left.\begin{array}{l}(14) \sim (\mathrm{I});\\ (15) \sim (\mathrm{III});\\ (16) \sim (\mathrm{V});\\ (17) \sim (\mathrm{VII});\end{array}\right\} \quad (\mathrm{A})$$

С другой стороны, формулам обобщенного машинного стиля, использованным в (А), соответствуют следующие команды «Минска-I»:

$$(\mathrm{I}) \sim (11),\ (12); \quad (\mathrm{V}) \sim (12);$$
$$(\mathrm{III}) \sim (11); \quad (\mathrm{VII}) \sim (13).$$

Анализируя эти соотношения, устанавливаем: 1) первые адреса команд «Киева» (IAK) соответствуют вторым адресам команд «Минска-I» (IIAM), если $'(\mathrm{IAK}) \neq S$, и модификациям 2-й или 3-й, в противном случае; 2) (IIAK) всегда соответствует (IAM); 3) (IIIAK) соответствует команда

2		IIIAK

если $'(\mathrm{IIIAK}) \neq S$, и пустой строке, в противном случае; 4) условию $'('\psi)_{\mathrm{I}} \neq 0$ соответствует условие ${}^{2}\psi \neq 0$, а условию $'('\psi)_{\mathrm{I}} = S$ — условие $'('\psi)_{\mathrm{II}} = 0$.

Учитывая эти соотношения, получаем БПАФ для «Минска-I»

$$B \ \ldots \ r \Rightarrow \chi;\ k \Rightarrow \psi;\ \gamma \oplus 1 \Rightarrow \Sigma$$
$$H \ \ldots \ \boldsymbol{Ц}\{'\varphi_0,\ C\varnothing \Rightarrow \varphi\}$$
$$np \cdot {}^{2}\varphi$$
$$(\ldots\ '\psi \Rightarrow '\Sigma;\ '\Sigma \oplus 1 \Rightarrow \Sigma$$
$$\boldsymbol{P}\{{}^{2}\varphi \neq 0\} \downarrow a$$
$$\boldsymbol{P}\{'('\psi_{\mathrm{II}} = 0\}\ '\chi \oplus 1 \Rightarrow \chi$$
$$'\chi \Rightarrow ('\psi)_{\mathrm{I}}$$

$$\textbf{Ц}\{'\psi\,(-1)\,\textbf{P}\{^2\psi \neq 0\} \Rightarrow \psi\}$$

$$L \ \ldots$$

	2		$'\chi$	$\Rightarrow {}'\psi$

$$a \ \ldots \ \textbf{P}\{('\psi \oplus 1)_2 = 2\} \downarrow c$$

$$'\psi \oplus 1 \Rightarrow \psi; \ '\psi \Rightarrow {}'\Sigma \ominus 1; \ 3 \Rightarrow ('\psi)_2$$

$$b$$

$$c \ \ldots \ \textbf{P}\{('('\psi)_{11} = 0\}\, 3 \Rightarrow ('\psi)_2 \downarrow 2 \Rightarrow ('\psi)_2$$

$$b \ \ldots \ '\psi \ominus 1 \Rightarrow \psi$$

$$H$$

$$) \ \ldots \ '\Sigma \ominus 1 \Rightarrow \Sigma; \ ''\Sigma \Rightarrow \psi$$

$$H$$

$$O \ \ldots \ ^2\varphi \Rightarrow ('\psi)_0$$

$$H$$

$$A \ \ldots \ \textbf{P}\{^2\psi = 0\}$$

	1		$^2\varphi$	$\Rightarrow {}'\psi \downarrow {}^2\varphi \Rightarrow ('\psi)_1$

Аналогично для «Урала-I» получаем следующий БПАФ:

$$B \ \ldots \ r \Rightarrow \chi; \ k \Rightarrow \psi; \ \gamma \oplus 1 \Rightarrow \Sigma$$

$$H \ \ldots \ \textbf{Ц}\{'\varphi_0; \ C\,\varnothing \Rightarrow \varphi\}$$

$$np \cdot {}^2\varphi$$

$$(\ldots \ '\psi \Rightarrow {}'\Sigma; \ '\Sigma \oplus 1 \Rightarrow \Sigma$$

$$\textbf{P}\{('('\psi \ominus 1) \neq 0\} \downarrow a$$

$$\textbf{P}\{('('\psi \ominus 1) = \textit{раб}\}\, '\chi \oplus 1 \Rightarrow \chi$$

	$'\chi$	$\Rightarrow {}'\psi$

$$\textbf{Ц}\{'\psi\,(-1)\,\textbf{P}\{^2\psi = 0\} \Rightarrow \varphi\}$$

$$L \ .\ .$$

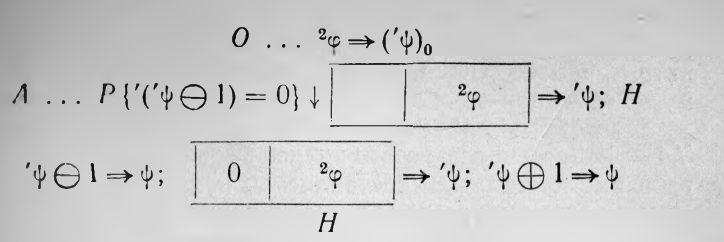
16	$'\chi$	$\Rightarrow {}'\psi$

$$a \ \ldots \ '\psi \ominus 1 \Rightarrow \psi$$

$$H$$

$$) \ \ldots \ '\Sigma \ominus 1 \Rightarrow \Sigma; \ ''\Sigma \Rightarrow \psi$$

$$H$$



4. ПРОГРАММИРУЮЩАЯ ПРОГРАММА ДЛЯ МАШИНЫ «УРАЛ-1», ОСНОВАННАЯ НА ПРИНЦИПЕ ПОЭЛЕМЕНТНОЙ РАСШИФРОВКИ АДРЕСНОГО АЛГОРИТМА ПП-УРАЛ-ЭРА[1].

Основные параметры программы

Принцип действия программирующей программы ПП-Урал-Эра основан на поэлементной расшифровке закодированной информации при однократном просмотре ее записи. Согласно сказанному ПП представляет собой совокупность алгоритмов, предназначенных для обработки (отображения в машинный код) отдельных представимых выражений входного языка, в качестве которых взяты элементарные символы или их последовательности.

Для удобства алгоритмы ПП названы соответственно отображаемым элементам: штрих-адрес, штрих-неадрес, скобка открывающая, скобка закрывающая, предикатная формула и т. д.; ниже приводится их описание в свободном адресном языке.

К началу работы ПП задаются следующие параметры: максимальная глубина скобок в записях адресных функций ($'h1$); максимальное число знаков операций в формулах засылки ($'h2$); максимальная глубина циклических скобок ($'h3$); количество передач управления строкам, находящимся в зонах, следуемых за ранее обрабатываемыми ($'h4$).

С целью упрощения ПП на свободный адресный язык наложены некоторые ограничения, сохраняющие его универсальность. В частности, допускаются формулы цикли-

[1] Работа выполнялась под руководством автора А. А. Гоба, В. Д. Рогач, Л. Г. Усенко и др. На начальной стадии в руководстве принял участие М. М. Бушко-Жук.

рования, вхождения, вычисляемого перехода и предикатные только специального вида; не допускаются формулы обмена и замены.

В частном случае при $'h1 = 12$, $'h2 = 40$, $'h3 = 7$, $'h4 = 12$ (все числа восьмиричные) максимальные участки рабочих программ, составляемых ПП в результате обработки одной зоны, не могут превышать 1500 команд.

Алгоритм ПП может быть разделен на два самостоятельных блока: 1) блок присвоения истинных адресов и распределения памяти; 2) блок программирования.

Для алгоритмов, заданных в истинных адресах с соблюдением ограничений, налагаемых стилем ПП, блок программирования может рассматриваться как вариант ПП. Приведем в адресном языке описание этого блока.

Блок программирования ПП (в дальнейшем для краткости будем называть его ПП) занимает 2140 ячеек оперативной памяти от нулевой ячейки по 2140, включая рабочие ячейки, массивы ПП и ее рабочее поле. Остальная часть оперативной памяти в процессе работы ПП используется следующим образом:

а) входная информация (обрабатываемый адресный алгоритм) — адреса от 2340 по 3000;

б) массив для хранения РП — адреса от 2200, а далее накрывается массив входной информации;

в) щит переключателя РП — адреса 3765 — 3777;

г) массив для хранения текущих значений параметров циклов — адреса 3755—3767;

д) массив для формирования передач управления на меченые строки (2141—2200).

Скорость работы ПП 50 команд в минуту (не считая ввода исходной информации). Обращение к вводным устройствам и к внешней памяти ПП может включать в РП только посредством нестандартных операторов.

ПП составляет программы в режимах фиксированной и плавающей запятой. Работа РП, составляемых ПП в плавающем режиме, основана на принципе моделирования. Для этого ПП формирует в РП команды обращения к специальным подпрограммам моделирования.

ПП реализует перевод адресного алгоритма в рабочую программу (РП) поформульно, т. е. путем обработки каждой отдельной адресной формулы в порядке их записи в алгоритме. Соответствующие данной адресной формуле команды РП формируются в рабочем поле ПП из

50_8 адресов [1], которое обозревается фиксатором (начало рабочего поля — адрес 1662). К началу обработки каждой формулы рабочее поле очищается и фиксатор ψ устанавливается в середине рабочего поля — на адресе 1720, по которому предварительно записывается код 02 0000.

Для записи РП отведена M-последовательность ($M = 2200$), называемая M-массивом хранения РП с фиксатором $N1$. В начале работы ПП M-массив очищен и $'N1 = 2200$. После обработки каждой адресной формулы команды РП переписываются по фиксатору $N1$. Исходная информация вводится в 2340-последовательность, обозреваемую фиксатором Φ.

Для больших программ адресный алгоритм разбивается на участки с учетом размерности $'\Phi$-последовательности, равной 440_8. Каждый участок записывается в отдельную зону. Выбор зон осуществляет ПП. По мере обработки отмеченных строк заполняется рабочий массив обработки меток, а именно: истинный номер первого приказа строки с меткой k засылается по адресу $\alpha \oplus k$. Передачи управления строкам, находящимся в еще не обработанных зонах, осуществляются с помощью щита переключателя R, представляющего собой массив из $'h4$ рабочих ячеек РП. В этих ячейках при обработке меток перехода на необработанные участки записываются условные коды меток, которые по мере обработки помеченных ими строк заменяются на истинные адреса команд. РП выдается на печать (перфоленту) по зонам. Щит R выдается на печать (перфорацию) вместе с рабочей программой последней зоны. Порядок размещения $'N1$ и $'\Phi$-последовательностей таков, что по мере обработки входной информации происходит наложение $'N1$-последовательности на $'\Phi$-последовательность.

Для программ, размещенных в одной зоне, можно непосредственно после составления РП без их вывода производить расчеты. Для этого:

1) на пульте управления набирается останов по адресу 1402;

2) по адресу N (0006) засылается 2200; это означает, что в качестве массива для размещения РП выбирается массив для хранения РП;

---

[1] Здесь и далее индекс 8 обозначает запись соответствующего числа в восьмеричной системе счисления.

3) после останова по адресу 1402 на места, занимаемые ПП, вводятся исходные данные к РП;

4) управление передается на начальную команду РП.

Кодирование

В результате работы блока присвоения истинных адресов и распределения памяти символам адресного языка ставятся в соответствие двухразрядные восьмеричные коды. Исключение составляют числа (целые) и адреса (истинные), под которые отводятся по четыре восьмеричных разряда. Таким образом, каждая короткая ячейка разделяется на три элементарных адреса. Элементарные адреса заполняются последовательно кодами элементов алгоритма. Знаки операций кодируются их машинными кодами; вводятся специальные коды:

N — начало адреса (код 17);

ц — модифицируемый адрес (код 20), т. е. адрес вида $a - {}'ц$,

где *ц* — параметр цикла;

НЧ — начало числа (код 40);

НО — нестандартный оператор (код 43);

М — метка передачи управления (код 31) и др. (табл. 4.).

После признака начала адреса (числа) в двух элементарных адресах записывается адрес (число). Первую половину адреса (числа) можно размещать в конце одной ячейки, вторую — в начале следующей (но обязательно одну за другой); *k*-ранг адреса кодируется соответствующей последовательностью кодов штрихов. После начала нестандартного оператора (*НО*) в следующем элементарном адресе указывается соответствующее восьмеричное число составляющих его команд, и непосредственно за ним с новой ячейки записываются эти команды. Никакого разделительного знака после *НО* не ставят.

Пустой код ПП пропускает; ставить его можно где угодно, но не разбивая адрес (число), а также не разделяя код штриха и следующего за ним признака адреса, код признака цикла и следующего за ним шага циклирования. Пустые коды можно использовать при исправлении ошибок кодирования.

В формуле вхождения метка подпрограммы и переключатель (ячейка возврата) кодируются без признаков адреса.

Таблица 4

Элементы адресного языка	Коды
АО (арифметические и логические операции)	от 1 до 14
Точка — признак конца формулы (.)	15
Признак адреса (*N*)	17
Признак модифицируемого адреса (*ц*)	20
Штрих (′)	21
Открывающая скобка (()	22
Закрывающая скобка ())	23
Засылка (→)	16
Начало предикатной формулы (*Р*)	26
Начало внешнего цикла (*Ц*1)	25
Начало внутреннего цикла (*Ц*2)	35
Конец цикла первого типа (*КЦ*1)	24
Конец цикла второго типа (*КЦ*2)	41
Метка (*М*)	31
Метки с троеточием (*М*...)	34
Начало формулы вхождения (*П*)	33
Признак адреса нулевого ранга (^0a)	40
Печать (*Пч*)	32
Операция «не равно по модулю» ($\mid \neq \mid$)	36
Конец зоны (*Я*)	27
Конец алгоритма (!)	30
Машинный останов (37 000)	37
Начало нестандартной формулы (*Нф*)	43
Пустой код	00

Символ начала плавающего режима (НПР) — код 44 — ставится после метки, отмечающей данную строку (если она имеется). После символа конца плавающего режима (КПР) — код 45 — никаких специальных разделительных знаков не ставится.

Входной язык

Входным языком ПП-Урал-Эра является некоторый стиль адресного языка. Перечислим налагаемые стилем ограничения.

1. Допускаются адресные функции, содержащие символы двухместных операций

арифметических $+$, $\mid-\mid$, $-$, $\times$, $:$,
логических (поразрядных) $\wedge$, $\vee$, $\cong$,

символ одноместной операции $'$ и адреса произвольного ранга. Нулевые ранги допускаются только для целых

чисел (положительных или отрицательных), являющихся первой компонентой любой из перечисленных двухместных арифметических операций.

Запись адресных функций строго скобочная. Допускаются лишние скобки, но это приводит к неэкономному программированию. По отношению ко всем двухместным операциям штрих-операция считается старшей. Скобки могут опускаться, если порядок выполнения операций совпадает с порядком их записи, а само выражение не находится в скобках.

Пример

Запись в свободном языке	*Запись в стиле входного языка*
$('a + 'b) : 'c2$	$('a + 'b) : 'c2$ или $'a + 'b : 'c2$
$2 + 'b \times 'c$	$2 + ('b \times 'c)$

Выражения адресных функций с символами одноместных операций во входном языке заменяются формулами вхождения соответствующих подпрограмм с засылкой результатов по некоторым адресам (стандартным ячейкам соответствующих подпрограмм) и записью адресных функций с учетом этих засылок.

Пример.

Свободный язык	*Входной язык*
$F('a + 'b) + 'a$	$'a + 'b \rightarrow d.$ $\boldsymbol{\Pi}\{'d\}\ f, g.$ $'c + 'a \ \ldots$

Здесь F — некоторая одноместная операция;
f — адрес начальной команды подпрограммы F;
g — переключатель (ячейка возврата с подпрограммы);
c — стандартная ячейка подпрограммы F ($'c$ — результат).

Для одноместных операций в фиксированном режиме, подпрограммы которых выдают результат на сумматоре, последняя строка во входном языке записывается с пропуском первой компоненты.

2. Формулы засылки записываются как выражения адресных функций, в которых знак засылки $\rightarrow$ рассматривается как символ двухместной операции того же старшинства. Поэтому правые части формулы засылки, содержащие символы двухместных операций, берутся в скобки.

Пример.

Свободный язык	*Входной язык*
$\ldots \to {}'a \oplus {}'b$	$\ldots \to ({}'a + {}'b)$
$\ldots \to {}'b \oplus 3$	$\ldots \to (3 + {}'b)$
$\ldots \to {}'a \oplus {}'a \otimes {}'b$	$\ldots \to ({}'a + ({}'a \times {}'b))$

3. Допускаются предикатные формулы вида

$$\boldsymbol{P}\{L\}\,\alpha \downarrow \beta,$$

где α и β — метки безусловного перехода,
L — логическое условие.

Из меток α и β только метка β может опускаться (если она — метка следующей строки).

Допускаются логические условия следующего вида:

$$A\theta B, \tag{5.1}$$

где A и B — произвольные адресные функции,
θ — один из символов операций отношения

$$\leqslant,\ |\leqslant|,\ \neq,\ |\neq|. \tag{5.2}$$

При записи логических условий символы выражения (5.2) рассматриваются как двухместные арифметические операции (того же старшинства). Таким образом: 1) правая часть (5.1), содержащая символы двухместных операций, берется в скобки; 2) не допускаются в качестве правой части (5.1) адреса нулевого ранга.

4. Допускаются формулы циклирования вида:

$$\boldsymbol{Ц}\{\alpha\,(\xi)\,0 \Rightarrow u\}, \tag{5.3}$$

или

$$\boldsymbol{Ц}1\,\{\,(\alpha\,(\xi)\,0 \Rightarrow u\}, \tag{5.4}$$

где α — адрес любого ранга, включая нулевой;
$\xi = -2$ или -1;
u — адрес из некоторого массива s, объем которого определяет допустимую максимальную глубину циклов.

Формулы (5.3) и (5.4) во входном языке записываются в виде:

$$\boldsymbol{Ц}\{\xi,\ \alpha\}, \tag{5.5}$$

$$\boldsymbol{Ц}1\,\{\xi,\ \alpha) \tag{5.6}$$

Символы ***Ц***{(***Ц***1{) соответствуют циклам, содержащим (не содержащим) в своей области действия другие циклы; (разделитель «,» и «}» не кодируются).

Имеется два способа обозначения концов циклов. Концы циклов, находящиеся в области действия некоторого заголовка цикла, за которыми внутри этой области следуют другие символы заголовков циклов, обозначаются через *КЦ*2. Все остальные концы циклов обозначаются через *КЦ*1.

На запись сложных циклических процессов, зависящих от параметров, во входном языке накладывается следующее ограничение. Параметр внешнего цикла не может входить (включая и формулу циклирования) в область действия внутреннего по отношению к нему цикла. Такая независимость параметров достигается на уровне входного языка путем соответствующих засылок. Поскольку во входном языке параметры циклов несовместимы, то введена сокращенная запись формул циклирования (формулы 5.5, 5.6). Соответственно этому в выражениях вида

$$a - {}'ц,$$

где a — адрес произвольного ранга;

$ц$ — параметр цикла, $-'ц$ рассматривается как единый специальный символ (код 20).

В конце областей действия формул циклирования ставится специальный знак КЦ. Для формул циклирования, отличных от (5.3), (5.4), можно строго сформулировать простые правила перевода их в допустимые формулы входного языка. Например, при $'b > 0$ формулу

$$\boldsymbol{Ц}\{'a\,('b)\,'c \Rightarrow \alpha\}\,M$$

следует заменить на

$$'a - 'b \Rightarrow \alpha$$
$$d \ldots {}'\alpha + 'b \Rightarrow \alpha$$
$$\boldsymbol{P}\{'\alpha \leqslant 'c\} \downarrow M,$$

где метка d не принадлежит множеству меток адресного алгоритма. В конце области действия формулы циклирования эта метка ставится как метка безусловного перехода (на начало цикла).

5. Формулы вхождения

$$\Pi\alpha\{a_1, a_2, \ldots; a_{i+1}, \ldots, a_n\}$$

во входном языке заменяются строками

$$\begin{gathered} a_1 \Rightarrow r_1 \\ a_2 \Rightarrow r_2 \\ a_{i-1} \Rightarrow r_{i-1} \\ \Pi\{a_i\}\alpha, \delta \\ \gamma \ldots {}'r_{i+1} \Rightarrow a_{i+1} \\ \vdots \\ {}'r_n \Rightarrow a_n, \end{gathered}$$

где r_j $(1 \leqslant j \leqslant n)$ — рабочие ячейки соответствующей подпрограммы; метка γ не принадлежит множеству меток адресного алгоритма и ставится обязательно.

δ — ячейка возврата подпрограммы, в которую ПП засылает метку γ. Формула $\Pi\{'a_i\}\alpha, \delta$ называется формулой вхождения во входном языке.

Для обращения к подпрограмме печати используется специальная формула вхождения, которая записывается в виде $Пч('a)$ и обозначает печать $'a$.

6. Нестандартный оператор (совокупность строк в машинном коде) просто переписывается ПП в соответствующее место рабочей программы.

7. Последовательность формул засылки, выполняющихся в режиме плавающей запятой, в языке берется в скобки плавающего режима, обозначенные символами **НПР** и **КПР** (коды 44 и 45).

По коду **НПР** ПП формирует команду обращения к блоку моделирования плавающего режима (**МПР**), а по коду **КПР** — уход с него.

Числа в плавающем режиме представляются в виде двоичных кодов, порядок и мантисса которых содержатся в одной полной ячейке; под порядок отведены разряды от 1-го по 6-й; 7-й разряд — знак порядка; под мантиссу отведены разряды с 8-го по 35, 36 разряд — знак мантиссы.

Все величины, содержащиеся внутри скобок **НПР** и **КПР**, воспринимаются как числа в плавающем режиме.

Поэтому внутри скобок **НПР** и **КПР** формирование адресов не допускается. Эти операции должны выполняться перед скобками. Например, выражение

$$\text{НПР}'('a + ('b + 'c)\ )\ldots$$

в языке не допускается; следует писать

$$'a + ('b + 'c) \Rightarrow \alpha$$
$$\text{НПР}''\alpha.$$

Фиксаторы ПП

Φ — фиксатор массива (последовательности) исходной информации;

$N1$ — фиксатор приказов РП (начальное значение $N1 = 2200$);

N — фиксатор истинных адресов приказов РП; начальное значение $'N$ определяется распределением памяти;

ψ — фиксатор формирования приказов в рабочем поле;

Σ — фиксатор массива рабочих ячеек ПП, в которых запоминаются адреса ячеек рабочего поля ПП при обработке открывающих скобок; содержимое адресов Σ-массива используется при обработке соответствующих закрывающих скобок;

χ — фиксатор массива рабочих ячеек РП, адрес первой рабочей ячейки $x0$ задается распределением памяти: $'\chi 0 = x0$.

$K1$ — фиксатор нижней занятой приказами РП ячейки рабочего поля; начальное значение $'K1$ произвольно;

$\psi 2$ — фиксатор первой верхней свободной ячейки ψ-массива; начальное значение $'\psi = 1717$;

$M1$ — фиксатор первой свободной нижней ячейки ψ-массива, используется при обработке предикатной формулы с операцией $| \neq |$;

B — фиксатор, используемый при обработке модифицированных адресов;

W — фиксатор массива для хранения истинных адресов начала и конца нестандартного опе-

ратора, используемый также при обработке формул вхождения;

ε — фиксатор массива, в котором запоминаются адреса для формирования команд передачи управления по признаку конца цикла;

s — фиксатор массива РП, используемого для хранения текущих значений параметров циклов;

$s1$ — фиксатор массива ПП, в котором запоминаются шаги циклирования; содержимые этого массива используются при обработке концов циклов.

Рабочие ячейки ПП

λ — используется для определения места записи приказа при обработке арифметических (логических) операций, засылки и предикатных формул; начальное значение $'\lambda = 0$;

La — ячейка для полуавтоматического распределения памяти; ПП осуществляет сдвиг всей информации для РП, заданной в условных адресах, на величину содержимого этого адреса; в начале работы с ПП по адресу La засылается константа сдвига, если она не равна нулю;

$r0$, $r00$ — рабочие ячейки ПП; вначале $'r0 = 'r00 = 0$.

π — счетчик элементарных адресов ячеек Φ-массива; начальное значение $'\pi = 0$;

φ — служит для выделения содержимого элементарного адреса Φ-массива; вначале $'\varphi = 0$;

K — счетчик для выделения полного адреса из Φ-массива; вначале $'k = 0$;

L — ячейка, по которой засылается полный выделенный адрес из Ф-массива; вначале $'L = 0$;

m — используется при обработке признака формулы вхождения; начальное значение $'m = 0$;

τ — используется при обработке признака предикатной формулы; начальное значение $'\tau = 0$;

з — ячейка, указывающая номер зоны обрабатываемой информации;

$N0$ — хранит номер первого приказа РП.

Константы, используемые ПП

i — первый адрес s-массива,

q, α — для выделения элементарного адреса.

$770000 = '(q \oplus 0)$
$007700 = '(q \oplus 1)$
$000077 = '(q \oplus 2)$
— для выделения соответственно содержимого I, II, III элементарных адресов

$004014 = '(\alpha \oplus 0)$
$004006 = '(\alpha \oplus 1)$
$004000 = '(\alpha \oplus 2)$
— для сдвига I, II, III элементарных адресов

$007777 = 'r$ — для выделения адресной части команды

$003776 = '\delta 0$ — для гашения знака.

Переключатели ПП

$Q1$ используется в алгоритме выделения элементарного адреса.

В процессе работы ПП $'Q1$ принимает значения:

1) $'Q1 = 3$, означающее переход на проверку условия «$'\varphi$ — арифметическая (логическая) операция»;

2) $'Q1 = 5$ — переход на алгоритм выделения адреса;

3) $'Q1 = 7$ — переход на обработку знака «штрих»;

4) $'Q1 = 33$ — переход на обработку символа $\Rightarrow$;

5) $'Q1 = 37$ — переход на обработку метки передачи управления;

6) $'Q1 = 57$ — переход на обработку модифицируемого адреса в формуле вхождения;

7) $'Q1 = 73$ — переход на обработку нестандартного оператора. Начальное значение $'Q1 = 3$.

$Q2$ используется в алгоритме обработки штрих-адреса. Начальное значение $'Q2 = 20$. Возможные значения $'Q2$:

1) $'Q2 = 20$ при обработке штрих-адреса;

2) $'Q2 = 21$ при обработке адреса нулевого ранга.

$Q3$ используется при обработке штрих-адреса. Возможные значения $Q3$;

1) $'Q3 = 24$ — при обработке адреса нулевого ранга;

2) $'Q3 = 26$ — при обработке начала циклов.

$Q4$ используется в блоке обработки символа засылки.

1) $'Q4 = 11$, если следующий за знаком $\Rightarrow$ элемент «штрих»;

2) $'Q4 = 36$, если этим элементом является символ «(».

$Q5$ используется для вывода на печать:

1) $'Q5 = 75$ — при обработке признака конца зоны;

2) $'Q5 = 74$ — при обработке признака конца исходной информации.

$M0$ используется при обработке формул вхождения:

1) $'M0 = 54$ — выход на выделение аргумента подпрограммы;

2) $'M0 = 60$ — уход на запись команды засылки в ячейку возврата;

3) $'M0 = 59$ — выход на формирование команды обращения к подпрограмме.

Описание алгоритмов ПП

Согласно принятому принципу поэлементной расшифровки входной информации ПП-Урал-Эра имеет звездную структуру. Центральным является алгоритм ЦА выделения элементарного адреса, распознавания символа, содержащегося в нем, и перехода в зависимости от последнего на соответствующий алгоритм. ПП вырабатывает дополнительную информацию, которую хранит в своих рабочих ячейках, массивах и переключателях при обозревании некоторых элементов:

а) символов «с тенью», т. е. влияющих на ход обработки непосредственно следующих за ними символов (например, штриха, влияющего на обработку следующих за ним (A, $'$,); в этом случае ПП подготавливает нужным образом соответствующие переключатели;

б) парных символов — скобок (,) и циклических скобок ***Ц***, КЦ; для каждого типа скобок ПП имеет специальные массивы рабочих ячеек для занесения определенной информации при обработке (, ***Ц***, которая используется затем при обработке соответствующих символов закрывающих скобок.

Перейдем к описанию алгоритмов ПП. При этом машинные коды в записях будем брать в прямоугольные рамки; «*сдвиг*» означает операцию сдвига кода левой компоненты на число разрядов, указанное правой компонентой, влево или вправо в зависимости от знака последней. $'f_0$ и $'f_a$ обозначают соответственно кодовую и адресную части содержимого адреса f.

Алгоритм ЦА

$$ЦА \ldots 0 \Rightarrow \pi;\ {}'\Phi \oplus 1 \Rightarrow \Phi$$
$$\boldsymbol{P}\{{}'\Phi > \Phi_{\max}\}!$$
$$2 \ldots ({}^2\Phi \wedge {}'({}'q \oplus {}'\pi))\ \textit{сдвиг}\ {}'(\alpha \oplus {}'\pi) \Rightarrow \varphi;\ {}'Q1$$
$$3 \ldots \boldsymbol{P}\{{}'\varphi \leqslant 14\}\, O2 \downarrow {}'(\theta \oplus {}'\varphi)$$
$$11 \ldots {}'\pi \oplus 1 \Rightarrow \pi$$
$$\boldsymbol{P}\{{}'\pi \leqslant 2\}\, 2 \downarrow ЦА$$

Здесь первая строка означает подготовку к выбору содержимого первого элементарного адреса по следующему адресу в исходной информации.

Предикатная формула во второй строке проверяет (в целях контроля) выход за пределы Φ-массива.

Строка с меткой 2 — засылка содержимого элементарного адреса по φ; $Q1$ — переключатель; исходное значение ${}'Q1 = 3$. В алгоритмах обработки символов «с тенью» ПП по адресу $Q1$ засылает соответствующую метку. По следующим трем строкам осуществляется распознавание символа ${}'\varphi$ и переход на соответствующий алгоритм ПП. Алгоритмы обработки элементарных символов заканчиваются переходом на метку 11 (последние две строки алгоритма $ЦА$) — подготовку выбора следующего элементарного адреса.

Алгоритмы обработки скобок „(„ и „)“

Смысл работы этих алгоритмов состоит в замене адресного выражения, стоящего в скобках, рабочей ячейкой, в которой будет получено соответствующее значение.

$$(\ldots 1 \oplus {}'\Sigma \Rightarrow \Sigma$$
$$\boldsymbol{P}\{{}'\Sigma > {}'\Sigma\max\}!$$
$$\boldsymbol{P}\{{}'\lambda = 0\}\, 12 \downarrow {}'\lambda \Rightarrow {}'\Sigma;\ 0 \Rightarrow \lambda;\ 13$$
$$12 \ldots {}'\psi \oplus 1 \Rightarrow {}'\Sigma$$
$$13 \ldots \boldsymbol{P}\{{}'\psi 2 \ominus {}'\psi = 1\}\, 18$$
$$14 \ldots {}'\chi \ominus 2 \Rightarrow \chi;\ {}^2\psi \oplus {}'\chi \Rightarrow {}'\psi;\ {}'\psi 2 \Rightarrow \psi$$
$$15 \ldots \boxed{16\,0000} \oplus {}'\chi \Rightarrow {}'\psi;\ {}'\psi \ominus 1 \Rightarrow \psi$$
$$16 \ldots 0 \Rightarrow \psi;\ {}'\psi \ominus 1 \Rightarrow \psi;\ \boxed{02\,0000} \Rightarrow {}'\psi$$

$$17 \ldots {}'\psi \ominus 1 \Rightarrow \psi 2;\ 11$$

$$18 \ldots \boldsymbol{P}\{{}^{2}\psi = \boxed{\boxed{30\,0000}}\}\ 19 \downarrow 16$$

$$19 \ldots {}^{2}\psi \oplus {}'\chi \Rightarrow {}'\psi;\ 15.$$

Обрабатывая открывающую скобку, ПП фиксирует (по ${}^{1}\Sigma$) адрес рабочего поля, по которому осуществляется возврат при обработке соответствующего символа закрывающей скобки.

Первая строка алгоритма подготовляет в рабочем Σ-массиве адрес; вторая строка проверяет превышение максимальной допустимой глубины скобок; дальнейшая группа команд, включая строку с меткой 12, засылает по адресу ${}^{1}\Sigma$ номер ячейки рабочего поля, по которому будет установлен фиксатор формирования приказов (ψ) при обработке соответствующей открывающей скобки. Кроме массива Σ установку фиксатора ψ может определять адрес λ. По этому адресу при обработке штриха также засылается адрес рабочего поля, на формирование команды в котором осуществляется возврат при обработке символов арифметических (логических) операций. По строке с меткой 13 осуществляется проверка возможности экономии команд и рабочих ячеек (переход по метке 18) за счет использования содержимого сумматора. Строка с меткой 14 подготовляет адрес следующей рабочей ячейки, оканчивает формирование приказа, записывая в его адресную часть адрес рабочей ячейки, и смещает фиксатор ψ на первую сверху свободную ячейку рабочего поля ПП (по фиксатору $\psi 2$).

По строкам от метки 15 до метки 17 подготовляется засылка результата вычисления, стоящего в скобках, в рабочую ячейку и устанавливается фиксатор ψ на адресе, в котором будет формироваться команда.

Строка с меткой 17 фиксирует адрес первой сверху свободной ячейки (по ${}'\psi 2$). По метке 11 происходит переход на ЦА.

$$) \ldots {}^{2}\Sigma \Rightarrow \psi;\ {}'\Sigma \ominus 1 \Rightarrow \Sigma;\ 11.$$

Первая формула осуществляет возврат фиксатора ψ на то место рабочего поля, с которого был совершен уход по соответствующей открывающей скобке и где начнется формирование команды. Вторая из формул осуществляет

сдвиг фиксатора Σ вверх, поскольку информация, выработанная при обработке соответствующего парного символа открывающей скобки, использована; после этого осуществляется переход на ЦА.

Алгоритм обработки штрих-адреса, штрих-неадреса и числа

Алгоритмы штрих-адреса и штрих-неадреса имеют следующую общую часть

$$\textit{штрих} \ldots 7 \Rightarrow Q1;\ 11$$
$$7 \ldots \boldsymbol{P}\{'\varphi = 17\}\ \textit{штрих-адрес}$$
$$\boldsymbol{P}\{'\pi = 0\} \downarrow '\pi \ominus 1 \Rightarrow \pi;\ \textit{штрих-неадрес}$$
$$'\Phi \ominus 1 \Rightarrow \Phi;\ 2 \Rightarrow \pi;\ \textit{штрих-неадрес}$$

Штрих — элемент «с тенью». Данный алгоритм подготовляет проверку условия: следующий элемент — адрес или неадрес и возвращает на ЦА, после чего по переключателю $Q1$ совершается переход на распознавание следующего символа (метка 7). Если этот символ — признак адреса, то совершается переход по метке 4 на алгоритм выделения адреса. В противном случае фиксаторы Φ и π, обозревающие исходную информацию, возвращаются в предыдущее положение и осуществляется переход на алгоритм штрих-неадрес.

Алгоритмы обработки числа и штрих-адреса имеют общую часть. При обработке числа предварительно выполняется строка

$$\text{число} \ldots 5 \Rightarrow Q1;\ 21 \Rightarrow Q2;\ 11$$

а при обработке штрих-адреса строка

$$5 \Rightarrow Q1;\ 11.$$

Здесь первая засылка подготовляет к выделению числа; вторая засылка подготовляет переключатель $Q2$ для перехода на формирование команды посылки числа на сумматор

$$4\ldots\ \textit{штрих-адрес} \ldots 5 \Rightarrow Q1;\ 11$$
$$5\ldots\ 'L\ \textit{сдвиг}\ (-6) \oplus '\varphi \Rightarrow L;\ 1 \oplus 'K \Rightarrow K$$
$$\boldsymbol{P}\{'K = 2\}\ 0 \Rightarrow Q1;\ 0 \Rightarrow K \downarrow 11$$
$$\boldsymbol{P}\{'m = 1\}\ 53 \downarrow 'Q2$$

$$20\ \ldots\ 'La \oplus {}''\psi \oplus 'L \Rightarrow '\psi;\ 23$$

$$21\ \ldots\ \boldsymbol{P}\{'('\psi \oplus 1)_0 \neq 25\}\ 22$$

$$0 \Rightarrow '\psi;\ 1 \oplus '\psi \Rightarrow \psi;\ 1 \oplus '\psi 2 \Rightarrow \psi 2;\ 22a$$

$$22\ \ldots\ \overline{|\ 20\,0000\ |} \oplus 'L \Rightarrow '\psi;\ 23$$

$$22a\ \ldots\ ''\psi \oplus 'L \Rightarrow '\psi$$

$$23\ \ldots\ 20 \Rightarrow Q2;\ 0 \Rightarrow L;\ 'Q3$$

$$24\ \ldots\ \boldsymbol{P}\{'('\psi \oplus 1) = 0\}\ '\psi \oplus 1 \Rightarrow \lambda$$

$$25\ \ldots\ 11$$

$$26\ \ldots\ \beta \oplus 'N \ominus '\psi 2 \Rightarrow '\varepsilon;\ 24 \Rightarrow Q3;\ 11.$$

Первая строка алгоритма штрих-адреса подготавливает переход к выделению адреса. Следующие три строки осуществляют выделение адреса, засылку нуля в $Q1$ (так как адрес не является элементом «с тенью») и восстановление счетчика K.

Далее проверяется условие, является ли выделенный адрес (число) аргументом подпрограммы ($'m = 1$), и совершается переход на дальнейшую обработку формулы вхождения, или по переключателю $'Q2$ на обработку адреса (на строку с меткой 20), или на обработку числа (строка с меткой 21).

В строке с меткой 20 формируется команда ($'L$ — адрес команды; $''\psi$ — код команды) с учетом возможного сдвига адресной части на $'La$ (по адресу La задается константа сдвига адресов исходных данных для РП).

В строке с меткой 21 проверяется условие, является ли данное число максимальным значением параметра цикла; если условие выполняется, то фиксаторы ψ и $\psi 2$ подготавливаются к записи этого числа в адресную часть команды цикла. В противном случае подготавливается команда засылки этого числа на сумматор (строка с меткой 22).

В строке с меткой 23 восстанавливаются начальные значения $Q2$ и L и по переключателю $'Q3$ совершается переход на метку 26 при обработке адреса (числа), относящегося к заголовку цикла, или на метку 24 в противном случае.

Строками от метки 24 до 25 осуществляется запоминание адреса следующей ячейки рабочего поля по адресу λ, если она свободна, и уход на ЦА.

С помощью строки с меткой 26 вычисляется и запоминается адрес, на который передается управление по концу цикла, восстанавливается исходное значение переключателя $Q3$ и совершается переход на ЦА.

$$\textit{штрих-неадрес} \ \ldots \ \boldsymbol{P}\{'m = 1\}\,28$$
$$\boldsymbol{P}\{'\psi_0 = 30\}\,28$$
$$\boldsymbol{P}\{'('\psi \oplus 1) = 0\}\,'\psi \oplus 1 \Rightarrow \lambda \downarrow 28$$
$$\boldsymbol{P}\{'\psi \ominus '\psi 2 = 1\}\,30$$
$$28 \ \ldots \ '\psi \ominus 1 \Rightarrow \psi$$
$$29 \ \ldots \ '\psi \ominus '\psi 2 \Rightarrow r0$$
$$\boldsymbol{Ц}\{'r0\,(\ominus 1)\,0 \Rightarrow C\}$$
$$'('\psi \oplus 1 \ominus 'C) \Rightarrow '\psi \ominus 'C$$
$$\boxed{30\,0000} \Rightarrow '\psi$$
$$'\psi 2 \ominus 1 \Rightarrow \psi 2;\ 0 \Rightarrow Q1;\ 11$$
$$30 \ \ldots \ \boxed{02\,0000} \Rightarrow '\psi;\ '\psi \Rightarrow B;\ 28.$$

По первым двум строкам этого алгоритма проверяется, программируется ли формула вхождения и предшествовал ли данному элементу штрих символ штрих. При выполнении этих условий (переход по метке 28) фиксатор ψ сдвигается на один адрес вверх и подготавливается сдвиг вверх составленной части РП, размещенной выше адреса $'\psi \oplus 1$. Число сдвигаемых команд вычисляется по адресу $r0$.

По третьей предикатной формуле проверяется условие, свободен ли адрес $'\psi \oplus 1$, и если оно выполняется, то этот адрес запоминается по адресу λ ($'\lambda$ используется при обработке символа открывающей скобки, следующего непосредственно за символом штриха).

Далее, как и в алгоритме открывающей скобки, проверяется возможность экономии команд, которая осуществляется по строке с меткой 30, где подготавливается кодовая часть команды вызова на сумматор и адрес $'\psi$ запоминается по адресу B ($'B$ используется при обработке модифицируемых адресов).

При невозможности экономии команд осуществляется переход на сдвиг РП (строка с меткой 28).

Алгоритм обработки символов (арифметических и логических) операций

$$O2 \ldots \boldsymbol{P}\{'\lambda \neq 0\}\,'\lambda \Rightarrow \psi;\ 0 \Rightarrow \lambda$$
$$'\varphi\ \textit{сдвиг}\ (-12) \Rightarrow {}'\psi;\ '\psi \Rightarrow B;\ 11.$$

В первой строке проверяется условие, фиксируется ли по адресу λ адрес ячейки рабочего поля, и при выполнении его на этот адрес сдвигается фиксатор ψ, а по адресу λ засылается нуль.

В следующей строке выделенный код операции засылается по адресу $'\psi$; $'\psi$ запоминается по адресу B. По метке 11 совершается переход на ЦА.

Алгоритм обработки признака конца формулы *тчк*

$$\textit{тчк} \ldots -1 \oplus 'K1 \ominus ''\psi \Rightarrow r0$$
$$\boldsymbol{Ц}\{'r0\,(-1)\,0 \Rightarrow C\}\,M1$$
$$'('K1 \ominus 'C) \Rightarrow {}'N1;\ 'N1 \oplus 1 \Rightarrow N1;\ 'N \oplus 1 \Rightarrow N$$
$$0 \Rightarrow 'K1 \ominus 'C$$
$$M1 \ldots \overline{\left|\ 020000\ \right|} \Rightarrow \beta;\ '\beta \Rightarrow B;\ '\beta \ominus 1 \Rightarrow \psi 2$$
$$\beta \Rightarrow \psi;\ 0 \Rightarrow \lambda;\ 'X0 \Rightarrow X;\ 11.$$

В первой строке вычисляется число адресов рабочего поля, занятых командами РП. Далее с помощью цикла эти команды пересылаются в массив хранения РП и одновременно сдвигаются фиксаторы хранения РП и ее истинных адресов, а также очищается рабочее поле.

Строки от метки $M1$ восстанавливают исходное значение фиксаторов и рабочих ячеек ПП.

Алгоритм обработки символа засылки

Этот алгоритм осуществляет обработку символа засылки и следующих за ним символов открывающих скобок и штрихов вплоть до признака адреса (числа)

$$\Rightarrow \ldots 33 \Rightarrow Q1$$
$$\boldsymbol{P}\{'\lambda = 0\}\,'\psi \Rightarrow \lambda;\ 11$$
$$31 \ldots {}'\lambda \Rightarrow {}'\psi;\ 11$$

$$33 \ldots \boldsymbol{P}\{'\varphi = '\}11 \Rightarrow Q4;\ 35$$

$$\boldsymbol{P}\{'\varphi = (\}36 \Rightarrow Q4;\ 35$$

$$\boxed{16\,0000} \Rightarrow '\psi;\ '\psi \Rightarrow B;\ '\psi \Rightarrow K1;\ 5 \Rightarrow Q1;\ 31$$

$$35 \ldots \boxed{30\,0000} \Rightarrow '\psi;\ '\psi \oplus 1 \Rightarrow '\psi;\ 'Q4$$

$$36 \ldots \boxed{16\,0000} \Rightarrow '\psi;\ '\psi \Rightarrow K1;\ '\Sigma \oplus 1 \Rightarrow \Sigma;$$

$$1 \oplus '\psi \Rightarrow '\Sigma;\ '\lambda \Rightarrow \psi;\ 0 \Rightarrow \lambda;\ 14.$$

В первой строке подготавливается возврат на алгоритм засылки по переключателю $Q1$. В следующей строке проверяется условие, фиксируется ли по λ адрес рабочего поля, и, если это так, по этому адресу рабочего поля устанавливается фиксатор ψ и совершается переход на ЦА.

В следующих двух строках в зависимости от элемента, следующего непосредственно после символа $\Rightarrow$, осуществляется подготовка переключателя $Q4$ и, если элемент является открывающей скобкой или штрихом, — выход на строку с меткой 35. В последней по адресу $'\psi$ формируется кодовая часть команды повышения ранга адреса, сдвигается на один адрес вниз фиксатор ψ и передается управление по $Q4$, т. е. на ЦА, если $'\varphi = '$, или на метку 36, если $'\varphi = ($.

В строке с меткой 36 по адресу $'\psi$ формируется кодовая часть команды «засылка по адресу». Обработка адресной части этой команды вначале незначительно отличается от соответствующего этапа работы алгоритма открывающей скобки, а переход на их общую часть осуществляется по метке 14.

Первый признак адреса (числа) после символа $\Rightarrow$ уводит из этого алгоритма. Предварительно (шестая строка) также формируется кодовая часть команды «засылка по адресу» в ячейке рабочего поля $'\psi$, адрес которой запоминается в B (на случай модификации) и в $K1$ ($K1$ — фиксатор нижней занятой приказами РП ячейки рабочего поля), и переключатель $Q1$ подготавливается для перехода на алгоритм выделения адреса (числа).

Фиксатор ψ по $'\lambda$ устанавливается на ячейке рабочего поля, в адресную часть которой в дальнейшем записывается соответствующий адрес (число).

Алгоритм обработки меток с троеточием

Метка ... $43 \Rightarrow Q1;\ 11$

$43\ \ldots\ {}'N \Rightarrow {}'\varphi \oplus \alpha$

$\boldsymbol{Ц}\{{}'h4\,(-1)\,0 \Rightarrow C\}\,42a$

${}'({}'R \max \ominus {}'C\} \Rightarrow r0$

$\boldsymbol{P}\{{}'(\alpha \oplus {}'\varphi) = {}'r0\}\,42b$

$\boxed{22\,0000} \oplus {}'N \Rightarrow {}'R \max \ominus C$

$0 \Rightarrow Q1;\ 11$

$42b\ \ldots$

$42a\ \ldots\ 0 \Rightarrow Q1;\ 11.$

Первая строка по признаку метки подготавливает возврат на данный алгоритм для обработки метки. В строке с меткой 43 истинный номер первого приказа, соответствующего отмеченной формуле, запоминается в рабочем массиве ПП по адресу $\alpha \oplus {}'\varphi$ (здесь ${}'\varphi$ = метке, $01 \leqslant {}'\varphi \leqslant 77$). Далее в цикле распознается, имелись ли передачи управления на эту метку. Если такие передачи встречались, то вместо условного кода $\alpha \oplus {}'\varphi$ ставится команда

$$22\,0000 \oplus {}'N,$$

т. е. команда передачи управления с истинным адресом.

Алгоритм обработки предикатных формул

Этот алгоритм состоит из двух частей, отмеченных метками P и P1; первая обрабатывает предикатные формулы с отношением $\leqslant$, $|\leqslant|$, $\neq$, вторая — с отношением $|\neq|$.

Эти алгоритмы обрабатывают символы предикатных формул $\boldsymbol{P}$ и их значения — метки безусловной передачи управления. Условия, содержащиеся в предикатных формулах, как было отмечено при описании входного языка ПП, обрабатываются поэлементно соответствующими алгоритмами ПП; поэтому символы отношений кодируются как

арифметические операции. Исключение составляет символ, обрабатывающийся алгоритмом $P1$.

$$P1 \ \ldots \ \boldsymbol{P}\{'\lambda \neq 0\}\,'\lambda \Rightarrow \psi;\ 0 \Rightarrow \lambda\downarrow$$

$$60a \ \ldots \ \boxed{12\,0000} \oplus '\delta 0 \Rightarrow '\psi,\ '\psi \oplus 1 \Rightarrow \psi;\ '\chi \ominus 2 \Rightarrow \chi$$

$$'\chi \oplus \boxed{14\,0000} \Rightarrow '\psi;\ 1 \oplus '\psi \Rightarrow M1,\ '\psi 2 \Rightarrow \psi$$

$$\boxed{16\,0000} \oplus '\chi \Rightarrow '\psi;\ '\psi \ominus 1 \Rightarrow \psi;\boxed{12\,0000} \oplus '\delta 0 \Rightarrow '\psi$$

$$'\psi \ominus 1 \Rightarrow \psi;\ \boxed{02\,0000} \Rightarrow '\psi;\ '\psi \ominus 1 \Rightarrow \psi 2;\ 0 \Rightarrow Q1;\ 11.$$

В первой строке этого алгоритма совершается соответствующая установка фиксатора ψ, после этого формируются команды

02 0000 **
12 3776
16 'χ $\overline{\overline{*}}$
. . . .
12 3776 *
14 'χ

Знаками $*$ и $\overline{\overline{*}}$ отмечены адреса, на которых первоначально устанавливались фиксаторы ψ и $\psi 2$. На месте многоточия размещаются приказы программы, соответствующие левой части отношения. Адрес, отмеченный двумя звездочками, является последним из адресов, в которых размещается программа правой части предикатной формулы.

$$P \ \ldots \ 1 \Rightarrow \tau;\ 11$$

$$37 \ \ldots \ 38 \Rightarrow Q1;\ 11$$

$$38 \ \ldots \ \boldsymbol{P}\{'\lambda = 0\}\,39\downarrow '\lambda \Rightarrow \psi;\ 0 \Rightarrow \lambda;\ 40$$

$$39 \ \ldots \ \boldsymbol{P}\{'M1 = 0\}\,40\downarrow 'M1 \Rightarrow \psi;\ 0 \Rightarrow M1$$

$$40 \ \ldots \ \boldsymbol{P}\{'\tau = 0\}\,41\downarrow \boxed{21\,0000} \Rightarrow r0;\ 0 \Rightarrow \tau;\ 42$$

$$41 \ \ldots \ '\psi \oplus 1 \Rightarrow \psi;\ '\psi \Rightarrow K1,\ \boxed{22\,0000} \Rightarrow r0$$

$$42 \ \ldots \ (\alpha \oplus '\varphi) \oplus 'r0 \Rightarrow '\psi$$
$$\boldsymbol{P}\{'\psi \ominus '\psi 2 = 2\}\,\beta \Rightarrow \psi 2$$
$$0 \Rightarrow Q1;\ 11.$$

По первой строке этого алгоритма признак обработки предикатной формулы засылается по адресу τ, который затем используется для обработки значений предикатной формулы. При обработке признака метки передачи управления совершается переход на строку с меткой 37, где подготовляется переключатель $Q1$.

Далее (строки с метками 39, 40) после выделения алгоритмом ЦА метки фиксатор ψ устанавливается на ячейку рабочего поля, в которой далее формируется приказ передачи управления.

В строке с меткой 40 определяется характер передачи управления (условная или безусловная) и соответственно подготавливается содержимое адреса $r0$, с помощью которого в строке с меткой 42 формируется этот приказ путем прибавления к кодовой части истинного адреса первой команды формулы, отмеченной соответствующей меткой. Попутно изменяются содержимые рабочих ячеек ПП1 λ, $M1$, τ, $K1$.

Программирование второй метки предикатной формулы (если она имеется) осуществляется фактически как программирование метки безусловного перехода; это обеспечивается засылкой $0 \Rightarrow \tau$ (строка с меткой 40) при программировании первой метки (первая метка во входном языке не может опускаться).

Алгоритм обработки модифицируемых адресов

$$-''B \Rightarrow 'B;\ 0 \Rightarrow B;\ 11.$$

В этом алгоритме по признаку модифицируемости адреса (код 20) адресу присваивается признак модифицируемости.

Алгоритм обработки символов «начало заголовка цикла»

$$Ц\ \ldots 0 \Rightarrow D;\ \boxed{60\,3777} \Rightarrow {}'\psi \oplus 2;\ {}'s \oplus 1 \Rightarrow s;$$

$$\boxed{16\,0000} \oplus {}'s \Rightarrow {}'\psi \oplus 3$$

$$Ц1\ \ldots\ Ц2 \Rightarrow Q1;\ 11$$

$$Ц2 \ldots {}'\varphi \Rightarrow {}'s1;\ \boxed{00\,1723} \Rightarrow K1$$

$$Ц3 \ldots \boxed{30\,0000} \Rightarrow {}'\psi;\quad \boxed{25\,0000} \Rightarrow {}'\psi \oplus 1$$

$$\boldsymbol{P}\{{}^{2}s1 \neq 2\}\ Ц4$$

$$\boxed{25\,4000} \Rightarrow {}'\psi \oplus 1$$

$$\boldsymbol{P}\{{}'D = 0\}\ Ц4$$

$$\boxed{60\,7777} \Rightarrow {}'\psi \oplus 2$$

$$Ц4 \ldots {}'s1 \oplus 1 \Rightarrow s1;\ 26 \Rightarrow Q3;\ 1 \oplus {}'\varepsilon \Rightarrow \varepsilon$$

$$\boldsymbol{P}\{{}'\varepsilon = \varepsilon N\}\ !\downarrow 0 \Rightarrow Q1;\ 11$$

$$Ц5 \ldots 2 \oplus {}'\psi 2 \Rightarrow \psi 2;\ 2 \oplus {}'\psi \Rightarrow \psi;\ 0 \Rightarrow D;\ Ц1.$$

При обработке кода 25 символа заголовка внешнего цикла совершается переход на начало программы (метка Ц), а в случае кода символа заголовка внутреннего цикла — на строку с меткой Ц5. Запись алгоритмов обработки указанных символов отличается только этими первыми двумя строками.

По строке с меткой *Ц* по адресу *D* засылается 0 и формируются команды, дублирующие счетчик циклов. В строке с меткой Ц5 это дублирование опускается, подготовляются фиксаторы ψ и $\psi 2$ и засылается признак $+0$ по адресу *D*. Во второй снизу строке проверяется превышение максимально допустимой глубины циклов.

Для кода 25 формируется следующая группа команд

30 ...
25 ...
60 ...
16 *'s*, (А)

а для кода 35 команды

30 ...
25 ...

По строке с меткой *Ц*1 замыкается ключ *Q*1 для возврата на данную программу и совершается переход на ЦА для выделения шага параметра цикла. По строке с меткой *Ц*2 запоминается шаг параметра цикла по фиксатору *s*1 и по адресу *K*1 запоминается адрес ячейки

рабочего поля, в которой формируется последняя команда заголовка цикла.

Независимо от способа задания максимального значения параметра цикла ПП вначале заготавливает в рабочем поле команды группы (А): если это значение задано по адресу нулевого ранга, то в алгоритме обработки адреса (строка с меткой 21) первая строка этой группы стирается.

Далее в случае шага, равного двум, исправляются команды группы (А) для работы по полным ячейкам. При этом с помощью условия $'D \neq 0$ проверяется необходимость исправления команды дублирования счетчика циклов.

По строке с меткой *Ц*4 подготовляются фиксатор $s1$, переключатель $Q3$ для вычисления адреса начала области цикла в рабочей программе и фиксатор ε, по содержимому которого будет заслан этот адрес в алгоритме обработки штрих-адреса (числа).

Алгоритм обработки символов «конец цикла»

$$Ц6 \ldots 0 \Rightarrow r0;\ Ц8$$

$$Ц7 \ldots -1 \Rightarrow r0$$

$$Ц8 \ldots \boldsymbol{P}\{'s0 = 's\}\ Ц11$$

$$0 \to B;\ \boxed{\ 00\,1722\ } \to K1;\ 's1 \ominus 1 \Rightarrow s1$$

$$Ц8а \ldots \boxed{\ 24\,0000\ } \oplus ''\varepsilon \to '\psi;\quad '\varepsilon \ominus 1 \Rightarrow \varepsilon$$

$$\boldsymbol{P}\{'B \neq 0\}\ 11$$

$$\boxed{\ 30\,0000\ } \oplus 's \Rightarrow '\psi \oplus 1$$

$$\boldsymbol{P}\{^2s1 \Rightarrow 2\}\ Ц10$$

$$\boxed{\ 25\,7777\ } \Rightarrow '\psi \oplus 2$$

$$Ц9 \ldots 's \oplus 'r0 \Rightarrow s;\ 11$$

$$Ц10 \ldots \boxed{\ 25\,3777\ } \Rightarrow '\psi \oplus 2;\ Ц9$$

$$Ц11 \ldots 11 \Rightarrow B;\boxed{\ 1720\ } \Rightarrow K1;\ Ц8а.$$

Алгоритмы обработки символов конца цикла *КЦ*1 и *КЦ*2 отличаются только первыми строками. По коду 41 совершается переход на строку с меткой *Ц*6, в которой по адресу $r0$ подготовляется константа 0. Вход в алго-

ритм — по коду 24 на метку *Ц*7. В строке с этой меткой по адресу *r*0 засылается —1.

По условию $'s0 = 's$ (список циклических скобок исчерпан) проверяется, является ли обрабатываемый символ признаком конца цикла, не содержащегося внутри других циклов; для подобных циклов программирование восстановления счетчика циклов опускается и подготавливается несколько иначе содержимое адресов *B* и *K*1.

По строке с меткой *Ц*8*а* подготавливается последняя команда цикла. Этим заканчивается программирование в случае конца самого внешнего цикла (проверка условия $'B = 0$ и выход на *ЦА*).

Если список циклических скобок полностью не исчерпан, предварительно восстанавливается соответствующий счетчик циклов (в зависимости от шага).

Алгоритм обработки формул вхождения

П ... $54 \Rightarrow M0;\ 1 \Rightarrow m;\ 'N1 \Rightarrow 'W;$ 11

53 ... $'M0$

54 ... $^2\psi \oplus 'L \Rightarrow \psi;\ 59 \Rightarrow M0$

55 ... $57 \Rightarrow Q1$

56 ... $0 \Rightarrow L;$ 11

57 ... $\boldsymbol{P}\{'\varphi \neq N_{\text{изм}}\}$ 58

$-'\beta \Rightarrow \beta$ 4

58 ... $5 \Rightarrow Q1;$ 5

59 ... $\boxed{22\,0000} \oplus 'L \Rightarrow \beta \oplus 1;\ \boxed{20\,0000} \oplus$

$\oplus 'N \oplus 4 \Rightarrow '\psi 2 \ominus 1$

$'\psi 2 \Rightarrow \psi;\ '\psi 2 \ominus 2 \Rightarrow \psi 2;\ 60 \Rightarrow M0$

$4 \oplus 'N1 \Rightarrow 'W \oplus 1;\ \beta \oplus 1 \Rightarrow K1$

$'W \oplus 2 \Rightarrow W;$ 55

60 ... $\boxed{16\,0000} \oplus 'L \Rightarrow '\psi;\ 0 \Rightarrow m;$ 56.

Данный алгоритм по формуле вхождения $\boldsymbol{П}\{'a\}\delta, \gamma$ формирует группу команд вида

20 *T*
16 γ
02 *a*
22 δ.

По первой строке подготовляется переключатель $M0$ и содержимое адреса m, которое используется в алгоритме штрих-адреса при обработке аргумента подпрограммы, и по адресу $'W$ засылается содержимое фиксатора $N1$ — адрес начальной команды рабочей программы, соответствующей формуле вхождения, и совершается переход на *ЦА* (на выделение аргумента).

После выделения аргумента совершается возврат на строку с меткой 53 (четвертая строка алгоритма штрих-адреса). По строке с меткой 54 формируется команда засылки аргумента на сумматор и подготовляется переключатель $M0$. Далее подготовляется переключатель $Q1$, освобождается адрес L и совершается переход на *ЦА*.

В строке с меткой 57 проверяется условие, является ли аргумент подпрограммы модифицируемым, и если это так, то команде $'\beta$ присваивается признак модификации.

По строке с меткой 58 подготовляется переключатель $Q1$ для выделения полного адреса и совершается переход на формирование адреса.

Далее (строка с меткой 59) фромируются команды передачи управления на подпрограмму и посылки на сумматор адреса команды рабочей программы, на которую совершается переход после выполнения подпрограммы; подготовляются фиксаторы ψ, $\psi 2$ и переключатель $M0$; в ячейке $'W \oplus 1$ запоминается адрес команды перехода на подпрограмму и совершается переход на подготовку выделения ячейки возврата с подпрограммы.

В строке с меткой 60 в рабочем поле формируется команда 16γ (γ — ячейка возврата подпрограммы) и подготовляется переход на *ЦА*.

Алгоритм обработки признаков «конец зоны» и «конец исходной информации»

Конец зоны... 75... $Q \Rightarrow Q5$; $64a$

Конец информации... 74... $R2 \Rightarrow Q5$

$$64\ a \ldots \ \overline{|002200|} \Rightarrow \psi;\ 'W0 \Rightarrow W$$

$$65 \ \ldots \ \boldsymbol{P}\{{}^2W = '\psi\}'('W \oplus 1) \Rightarrow \psi;\ 'W \oplus 2 \Rightarrow W;\ 68$$

$$''\psi \wedge 'q \Rightarrow r0$$

$$\boldsymbol{P}\{'r0 = \overline{|210000|}\}\ 69$$

$$\boldsymbol{P}\{'r0 = \overline{\boxed{22\,0000}}\}\,69$$
$$67\ \ldots\ {}'\psi \oplus 1 \Rightarrow \psi$$
$$68\ \ldots\ \boldsymbol{P}\{'\psi \leqslant 'N1\}\,65 \downarrow 71$$
$$69\ \ldots\ {}^2\psi \wedge {}'r \Rightarrow r00;\ {}'r0 \Rightarrow {}'\psi$$
$$\boldsymbol{P}\{{}^2r00 = 0\}\,70$$
$${}^2r00 \oplus {}^2\psi \Rightarrow {}'\psi;\ 67$$
$$70\ \ldots\ {}'R \oplus 1 \Rightarrow R$$
$$\boldsymbol{P}\{'R > 'R\ \max\}!$$
$$'r00 \Rightarrow 'R;\ {}^2\psi \oplus {}'R \Rightarrow {}'\psi;\ 67$$
$$71\ \ldots\ 'N1 \ominus 1 \Rightarrow r0;\ {}'r0 - \overline{\boxed{00\,2200}} \Rightarrow r00$$
$$'N0\ \textit{сдвиг}6 \Rightarrow 0000$$
$$\boldsymbol{Ц}\{'r00\,(-1)\,0 \Rightarrow C\}\,КЦ$$
$$'('r00 \ominus 'C) \Rightarrow 0001$$
$$Пч'4000$$
$$КЦ\ \ldots$$
$$'Q5$$
$$R2\ \ldots\ R\ \max \Rightarrow r0;\ r0\ \textit{сдвиг}^{(-6)} \Rightarrow 0000$$
$$\boldsymbol{Ц}\{11\,(-1)\,0 \Rightarrow C\}\,КЦ1$$
$$'('R\ \max - 'C) \Rightarrow 0001$$
$$Пч'4000$$
$$КЦ1\ \ldots\ !$$
$$Q\ \ldots\ 002200 \Rightarrow N1$$
$$\boldsymbol{Ц}\{54\,(-2)\,0 \Rightarrow C\}$$
$$0 \Rightarrow 'W\ \max \ominus 'C$$
Очистка массивов $\ldots\ 0 \Rightarrow K1,\ Q1,\ Q4,\ Q2,\ Q5$
$$\boldsymbol{Ц}\,(11\,(-1)\,0 \Rightarrow C\}$$
$$0 \Rightarrow '\Sigma\ \max \ominus 'C$$
$1720 \Rightarrow \psi;\ 'N1 \Rightarrow N0;\ 1 \oplus 'з \Rightarrow з$ ввод $'з$ [ввод новой зоны].

Ввиду некоторой аналогии запись этих алгоритмов объединена. Обработка признака «конец зоны» начинается с первой строки, а признака «конец информации» — со второй. В этих строках подготавливается разветвление алгоритма по ключу $Q5$.

Начиная от метки $64а$ и далее работает алгоритм присваивания истинных адресов командам передачи управле-

ния; исключение составляют адреса тех команд, которые записаны в массиве W (адреса в этих командах уже истинные).

В этом алгоритме фиксатор ψ обозревает массив хранения рабочей программы. Согласно принятому способу заполнения W-массива всегда

$$^2W \geqslant {}'\psi.$$

Если

$$^2W = {}'\psi,$$

то осуществляются засылки, по которым пропускаются команды рабочей программы с адресами от $''W$ до $'('W + 1)$ как содержащие истинные адреса).

При

$$^2W > {}'\psi$$

для команд передач управления по адресу $r00$ засылается их адресная часть (условный адрес передачи управления). При наличии такой записи в массиве обработки меток ($^2r00 \neq 0$) условный код адреса заменяется истинным, в противном случае он записывается в R-массив, и по $'\psi$ в адресную часть команды передачи управления записывается адрес соответствующей ячейки R-массива. После каждого перехода на новый адрес в массиве хранения $РП$ проверяется условие конца этого массива (строка с меткой 67).

Начиная от строки с меткой 71, подготовляется и осуществляется печать команд $РП$ и по ключу $Q5$, в случае конца информации, печатается содержимое R-массива и работа алгоритма заканчивается. В случае конца зоны восстанавливаются фиксаторы, подготовляются рабочие ячейки и переключатели ПП, а также производится выборка информации следующей зоны.

Алгоритм обработки символа «печати»

Печать ... $\overline{\boxed{320000}} \Rightarrow {}'\psi \oplus 1;\ {}'\psi \oplus 1 \Rightarrow K1;\ 11;$

Алгоритм формирует команду печати. Засылка содержимого соответствующего адреса на сумматор осуществляется в блоке обработки штрих-адреса.

Алгоритм обработки символа останова

$$Останов\ \ldots\ \boxed{|370000|} \Rightarrow {}'\psi;\ {}'\psi \Rightarrow K1;\ 11$$

Алгоритм программирует команду останова. Для останова по адресу последний дописывается в адресную часть команды алгоритмом штрих-адреса.

Алгоритм обработки символов плавающего режима

При обработке этих кодов непосредственно в поле хранения *РП* формируются команды обращения к программе моделирования режима плавающей запятой и соответственного выхода с нее. 3734 — ячейка возврата, 3445 — адрес начала подпрограммы моделирования.

$$Начало\ плавающего\ режима\ \ldots\ 2 \oplus \boxed{|200000|} \oplus {}'N \Rightarrow {}'N1$$

$$\boxed{|163734|} \Rightarrow {}'N1 \oplus 1;\ \boxed{|223445|} \Rightarrow {}'N1 \oplus 2$$

$$3 \oplus {}'N1 \Rightarrow N1;\ 3 \oplus {}'N \Rightarrow N;\ 11.$$

$$Конец\ плавающего\ режима\ \ldots\ 0 \Rightarrow {}'N1;\ \ 1 \oplus {}'N1 \Rightarrow N1;$$

$$1 \oplus {}'N \Rightarrow N;\ 11$$

Моделирующая программа плавающего режима

Входной информацией для моделирующей программы (МП) являются приказы РП в фиксированном режиме. Результат работы МП — выполнение этих приказов в плавающем режиме.

Перед группой приказов, выполняемых в плавающем режиме, ставятся команды обращения к МП, которые имеют вид

К + 0	20	К + 2	
К + 1	16	3734	ячейка возврата
К + 2	22	3445	начало МП

При составлении рабочих программ программирующей программой эти приказы формируются ПП. Моделирующая программа состоит из блоков сложения, умножения, деления, вычитания, вызова числа на сумматор, засылки

по адресу, посылки числа в регистр, умножения в регистре, вычитания по модулю, формирования модифицируемого адреса. МП обрабатывает информацию в следующем порядке: выделяет код операции, затем адреса, определяет код операции, разделяет число на порядок и мантиссу, помещая мантиссу в полную, а порядок в короткую рабочие ячейки МП, и передает управление на соответствующий блок. После выполнения формулы результат засылается в рабочие ячейки МП $a1$ (мантисса) и $a2$ (порядок), моделирующие сумматор.

Обрабатывая приказ РП вида 16 c, МП передает управление на блок засылки числа. В ячейке $a1$ очищаются последние 7 разрядов для порядка и его знака. Порядок, содержащийся по адресу $a2$, сдвигается в младшие разряды, затем мантисса и порядок объединяются в одну полную ячейку и результат засылается по адресу c.

МП последовательно моделирует приказы до тех пор, пока не встретит код — признак конца плавающего режима.

Исходные данные для программ, имеющих обращение к программе МП, можно задавать в десятичной системе. Перед работой РП десятичные числа, обрабатываемые в плавающем режиме, переводятся в восьмеричные с помощью специальной программы перевода. Соответственно после выполнения расчетов восьмеричные числа из плавающего режима с помощью другой специальной программы переводятся в десятичные.

Переключатель θ

Для реализации формулы вычисляемого перехода (переключателя)

$$'(\theta \oplus '\varphi),$$

имеющегося в алгоритме ЦА, ПП содержит щит переключателя θ, состоящий из 24 адресов (по числу различаемых кодов), в которых записаны команды безусловного перехода на начальные строки соответствующих алгоритмов обработки тех или иных символов.

ГЛАВА VI

ПОСТРОЕНИЕ СХЕМ ОБОЗРЕВАНИЯ В КОДАХ КОНКРЕТНЫХ МАШИН

(Машина «Урал»)

В этой главе на примере универсальной ЭВЦМ «Урал» [16] продемонстрируем применение адресного языка для построения программ в машинных кодах. Кроме того, на том же примере покажем возможность использования адресного языка для описания математических параметров машин [2].

1. ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

Универсальная ЭВЦМ «Урал» имеет следующие эксплуатационно-технические параметры:

1. Системы счисления для чисел при вводе в машину и выводе на печать — десятичную и восьмеричную.
2. Разрядность восьмеричных чисел при вводе в машину и выводе на печать — 12 разрядов.
3. Разрядность десятичных чисел при вводе в машину и выводе на печать — 9 разрядов (младший разряд всегда четный).
4. Систему счисления для команд при вводе в машину и выводе на печать — восьмеричную.
5. Систему счисления для чисел и команд в машине — двоичную.
6. Разрядность чисел в машине — 36 двоичных разрядов (из них один знаковый).
7. Разрядность чисел, выводимых из машины на перфоленту — 36 двоичных разрядов (один знаковый).
8. Форму представления чисел в машине — с запятой, фиксированной перед старшим разрядом.
9. Управление — автоматическое с помощью программы и ручное с пульта управления.

10. Принцип работы основных устройств — параллельно-последовательный с фиксированной длительностью такта работы машины.

11. Ввод чисел и команд в машину — автоматический с накопителя на перфоленте и ручной с пульта управления.

12. Емкость накопителя на магнитном барабане — 1024 тридцатишестиразрядных (полных) двоичных кодов или 2048 восемнадцатиразрядных (неполных). Нумерация ячеек в машине восьмеричная. Последний номер ячейки в восьмеричной системе счисления 3777.

13. Накопитель на магнитной ленте (НМЛ):

а) емкость до 40 000 тридцатишестиразрядных двоичных кодов или до 80 000 восемнадцатиразрядных;

б) количество зон — 255;

в) скорость записи и воспроизведения — 4500 ± 10% чисел в минуту.

14. Накопитель на перфоленте (НПЛ):

а) емкость — до 10 000 чисел или команд;

б) количество зон — 127;

в) скорость ввода — 4500 ± 10% чисел в минуту.

15. Время выполнения отдельных операций:

а) все операции, кроме деления и нормализации,— 10 *мсек;*

б) нормализация — 20 *мсек;*

в) деление — 40 *мсек.*

16. Скорость печати результатов — 100 ± 10% чисел в минуту.

17. Скорость перфорации чисел от клавишного устройства 6—10 чисел в минуту.

18. Скорость перфорации чисел из машины 150 ± 10% чисел в минуту.

19. Скорость реперфорации чисел с ленты с помощью контрольно-считывающего устройства — 180 ± 10% чисел в минуту.

20. Скорость сравнения двух лент на контрольно-считывающем устройстве — 200 ± 10% чисел в минуту.

21. Полезное время работы — не менее 18 *ч* в сутки.

22. Режим работы машины возможен круглосуточный. Допустимы перерывы в решении задач с выключением машины.

23. Систему контроля — оперативный контроль с помощью текстпрограмм и профилактический контроль путем изменения режима работы схем.

24. Источник энергии — сеть трехфазного переменного тока напряжением 220 *в* ± 10% и частотой 50 *гц*.

25. Потребляемую мощность — 7,5 *квч* ± 10%.

26. Конструкцию — блочную.

2. ПРЕДСТАВЛЕНИЕ ЧИСЕЛ И ВНУТРЕННЯЯ ОПЕРАТИВНАЯ ПАМЯТЬ

В рассматриваемой машине реализовано два кода изображения двоичных чисел: прямой и обратный.

Для выполнения арифметических операций в прямом коде необходимо иметь два отдельных устройства: сложения и вычитания.

Обратный код позволяет вычитание заменить сложением чисел, представленных в обратном коде. В машине «Урал» сумматор работает в обратном коде. При этом сложение чисел, представленных в обратном коде, осуществляется по схеме циклического сложения, по которой перенос из старшего разряда, если он возникает, производится в младший разряд суммы. Перенос происходит, когда арифметическая сумма обратных кодов чисел больше или равна 2; в противном случае циклическое сложение совпадает с арифметическим.

Если сумма двух правильных дробей по абсолютной величине больше или равна единице, то в машине получается переполнение разрядной сетки и результат искажается. Чтобы зафиксировать момент переполнения разрядной сетки сумматора, в машине осуществлен модифицированный обратный код, при котором можно отличать правильные дроби от неправильных.

Как уже отмечалось, в простом обратном коде знак числа, выраженный цифрой 0 или 1, ставится в разряде целой единицы. Это исключает возможность изображать в обратном коде числа, большие или равные 1 по абсолютной величине.

В сумматоре «Урала» для знака числа введен второй разряд перед запятой, а первый разряд перед запятой — для целой части. Такой код называется *модифицированным обратным кодом*. В случае несовпадения цифр, стоящих в двух старших разрядах сумматора (что означает переполнение разрядной сетки), в машине вырабатывается сигнал переполнения φ.

«Урал» имеет оперативное запоминающее устройство —

накопитель на магнитном барабане (НМБ), который предназначен для записи, хранения и выдачи чисел и команд. НМБ состоит из магнитного барабана (МБ) и регистров, связанных с МБ и с другими устройствами машины. Разрядная сетка АУ содержит 37 разрядов, из которых 35 предназначены для дробной части числа.

Таким образом, возникает потребность записывать на МБ и считывать числа с точностью до 2^{-35}. Для этой цели в машине имеется возможность объединять две смежные ячейки магнитного барабана, из которых первая должна быть с четным номером, вторая — с нечетным. Такая ячейка, состоящая из двух восемнадцатиразрядных (или их еще называют короткими) ячеек, называется *длинной ячейкой*. Она состоит из 36 двоичных разрядов. Нумерация разрядов ведется справа налево, начиная с 1 и кончая 36, по степеням 2; 36-й разряд — знаковый, 35-й — 2^{-1}, ..., 1-й — 2^{-35}. Номер длинной ячейки есть номер соответствующей короткой четной ячейки, увеличенной на 4000. Например, две короткие ячейки 0004 и 0005 образуют одну длинную с адресом 4004.

Таким образом, содержимое двух неполных ячеек 0004 и 0005, записанное в разные такты, можно выдать в сумматор АУ за один такт по адресу 4004. Если в команде адрес ячейки больше или равен 4000, то при исполнении данной команды с таким адресом участвует полная (длинная) ячейка МБ.

Магнитный барабан может работать в следующих режимах:

1. Групповой переписи чисел по коротким ячейкам накопителя на перфоленте с НПЛ — режим переписи восьмеричных чисел (программы).

2. Групповой переписи чисел по длинным ячейкам с НПЛ и с НМЛ — режим переписи десятичных чисел в десятично-двоичном виде. Для этой цели в состав НМБ входит специальный блок групповых операций.

3. Чтения с МБ и выдачи в АУ и УУ чисел и команд по заданному адресу.

4. Записи на МБ по заданному адресу чисел и команд с АУ — одиночная запись.

5. Выборки команды по заданному адресу и выдача ее в УУ (в регистр команд *К*). Адрес команды надлежащей выборки выдает УУ (счетчик команд *С*).

Для всех этих режимов соответственно в НМБ

имеются специальные блоки. Одновременно в режиме записи и чтения НМБ работать не может.

Кроме этого, НМБ может работать в режиме чтения с МБ в контрольный регистр R для визуального наблюдения (контроля) содержимого любой ячейки МБ. В этом режиме НМБ работает совместно с другими режимами все время, за исключением режима одиночной записи.

Контрольный регистр R имеет на пульте сигнализации 36 сигнальных лампочек с надписью «контрольный регистр», а на пульте управления — 12 тумблеров с надписью «адрес контрольного регистра». При помощи этих тумблеров можно набрать номер любой ячейки магнитного барабана (адрес), содержимое которой будет читаться с МБ в R.

3. ПРОГРАММНЫЕ РЕГИСТРЫ

Рассмотрим программные регистры машины «Урал». В машине имеются следующие программные регистры:

1) S — сумматор АУ;
2) r — регистр АУ;
3) C — счетчик команд;
4) K — регистр команд; этот регистр разбивается на части:
   а) ξ_0 — регистр признака модификации (18-й разряд);
   б) K_0 — регистр кода операции (разряды 17—23);
   в) ξ_1 — регистр признака полноты ячейки (12-й разряд);
   г) K_1 — регистр адреса (разряды 1—11);
5) ω — триггер признака для условной передачи управления;
6) μ — регистр циклов;
7) φ — триггер переполнения;
8) D — дешифратор команд;
9) H — регистр «начальный пуск»;
10) R — контрольный регистр.

Роль программных регистров играют также некоторые ручные переключатели, в связи с чем их следует включить в общий список [1]:

11) χ_i ($i = 1, 2, \ldots, 7$) — ключи;
12) T_1 — тумблер «блокировка φ»;
13) T_2 — тумблер «остановка по φ».

[1] Здесь указаны лишь те программные регистры и ручные переключатели, которые нужны для описания устройств и набора операций машины.

Для визуального наблюдения на панели сигнализации имеется набор сигнальных лампочек, показывающих содержимое некоторых регистров машины.

Содержимое регистров может изменяться автоматически при выполнении команд программы или вручную через пульт управления ПУ. С этой целью на пульте помещены средства занесения информации в соответствующие регистры (тумблеры, кнопки, клавиши).

Сумматор *S* предназначен для выполнения арифметических и логических операций над числами и командами, представленными в двоичной системе счисления. В сумматоре числа могут храниться только в обратном коде. Сложение многоразрядных чисел, изображенных обратным кодом, сводится к их поразрядному сложению с учетом возникающих переносов из младших разрядов в старшие и с циклическим переносом из знакового разряда в младший разряд. Сумматор состоит из цепочки одноразрядных сумматоров (счетчиков), построенных на триггерных схемах, работающих по модулю 2.

Сумматор имеет 37 основных двоичных разрядов (которым на панели сигнализации соответствует система 37 сигнальных лампочек с надписью «сумматор», объединенных в группы по 3 лампочки для удобства обозрения и пронумерованных справа налево) и 6 дополнительных.

В 35 разрядах сумматора (с 1 по 35-й) помещается дробная часть числа. 36-й разряд отведен для целой части числа; 37-й — для знака числа. Запятая фиксируется между 35 и 36 разрядами. Максимальное по модулю число, которое может храниться в сумматоре, $2-2^{-35}$.

При нормальной работе (с фиксированной запятой) содержимое сумматора не должно превышать максимального числа, которое может храниться в ячейке магнитного барабана, т. е. числа $1-2^{-35}$.

Приведем таблицу положений 36-го и 37-го разрядов при различных содержимых сумматора

36	37	*S*
0	0	>0
1	1	<0
0	1	>1
1	0	<-1

Если в результате счета в сумматоре S получено число, по модулю большее, чем максимально возможное в ячейке магнитного барабана, что можно заметить по несовпадению содержимого 36-го и 37-го разрядов, то устройством управления вырабатывается управляющий сигнал, который помещается в триггер переполнения φ: $'\varphi = 1$ при переполнении, в противном случае $'\varphi = 0$.

Регистр r является промежуточной памятью АУ. В любом случае занесение чисел в сумматор S осуществляется через регистр r. Регистр имеет 36 основных двоичных разрядов. Старший, 36-й разряд регистра, — знаковый, а все остальные — разряды дробной части числа. Регистр r связан с панелью сигнализации 36 сигнальными лампочками, которые расположены под сигнальными лампочками сумматора АУ и имеют надпись «регистр АУ».

Регистр r имеет также связь со специальным регистром занесения чисел из устройства управления. Последний предназначен для занесения чисел в АУ (с учетом знака) с пульта управления.

Счетчик команд ***C*** служит для задания последовательности выполнения команд и изменения ее в ходе решения задачи. Содержимое счетчика команд может меняться на величину, отличную от 1, только с помощью команд передачи управления (коды 21, 22, 23, 24). Счетчик команд C состоит из 11 разрядов, которым на панели сигнализации соответствует 11 сигнальных лампочек с надписью «счетчик команд», по которым визуально можно определить содержимое C, т. е. номер команды, которая будет выполняться в следующем такте машины.

Минимальное число в восьмеричной системе, которое может храниться в C —0000; максимальное — 3777 (это соответствует допустимому многообразию адресов).

Регистр команд ***K*** служит для приема кода команды из накопителя на магнитном барабане, а также для алгебраического суммирования кодов команд при выполнении соответствующих операций (коды 25 и 30) без обращения к сумматору АУ. Он имеет 18 разрядов; на панели сигнализации имеется соответственно 18 сигнальных лампочек с общим наименованием «регистр команд» K.

Дешифратор команд расшифровывает код операции поступившей в него (выполняющейся) команды и выдает сигнал, по которому машина настраивается на выполнение данной операции. Код операции выполнившейся

команды сохраняется в дешифраторе команд D до поступления кода операции следующей команды. Таким образом, остановив в произвольный момент времени машину, по сигнализации от трех регистров («дешифратор команд D», «счетчик команд C» и «регистр команд K»), получаем: в счетчике команд C номер той команды, которая будет выполняться; в регистре команд K — команду, которая будет выполняться; в дешифраторе команд D — код операции команды, которая перед этим выполнялась. Таким образом, можно полностью определить место в программе (на барабане), на котором приостановлены вычисления.

4. ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ И НАБОР ЭЛЕМЕНТАРНЫХ ОПЕРАЦИЙ

Отдельный цикл работы машины «Урал» состоит из: 1) выполнения команды $'K$, т. е. команды, код которой хранится в данный момент в регистре команд; 2) засылки кода очередной команды в регистр команд. В зависимости от содержимого K_0 машина выполняет следующие операции.

Арифметические операции

1. $'K_0 = 01$ — сложение. По этой команде выполняются действия:

$$'S + '('K_1 \ominus '\xi_0 \cdot 'u) \Rightarrow S$$
$$\operatorname{sign} 'S \vee |'(K_1 \ominus '\xi_0 \cdot 'u)| \Rightarrow r$$
$$\boldsymbol{P}\{\operatorname{sign} 'S = 1\}\, 1 \Rightarrow \omega \downarrow 0 \Rightarrow \omega$$
$$\boldsymbol{P}\{|'S| > 1\}\, 1 \Rightarrow \varphi \downarrow 0 \Rightarrow \varphi$$
$$\boldsymbol{P}\{'\varphi = 0\}\, a$$
$$\boldsymbol{P}\{('T_1 = 0) \wedge ('T_2 = 0)\}\, 'C \oplus 1 \Rightarrow C;\ a \downarrow b$$
$$b \ \ldots\ \boldsymbol{P}\{'T_1 = 0\}!$$
$$a \ \ldots\ 'C \oplus 1 \Rightarrow C$$
$$^2C \Rightarrow K.$$

Если $'\xi_0 = 0$, т. е. признак изменяемости команды отсутствует, то выполняется засылка

$$'S \oplus {}^2K_1 \Rightarrow S.$$

Если же $'\xi_0 = 1$, то адрес модифицируется, т. е. уменьшается на величину содержимого регистра циклов. При этом, если $'\xi_1 = 0$, выбирается содержимое неполной

ячейки с адресом $'K_1$, если $'\xi_1 = 1$, — полной. В последнем случае $'K_1$ должно быть четным числом. В машине, как было указано в списке регистров, имеются регистр переполнения разрядной сетки φ и связанные с ним тумблеры T_1 и T_2.

При выполнении арифметических операций 01, 03, 04, 05, 06, 07; если $|'S| \geqslant 1$, в триггер φ засылается 1, а если $|'S| < 1$, в φ засылается 0. При этом, если $'T_1 = 1$ (блокировка φ включена), то выполняется действие

$$'C \oplus 1 \Rightarrow C$$

(что означает переход к следующей команде программы); если $'T_1 = 0$, $'T_2 = 1$, то машина останавливается; если $'T_1 = 0$, $'T_2 = 0$, то выполняется действие

$$'C \oplus 2 \Rightarrow C,$$

т. е. пропускается одна команда.

Содержание строк, начиная с третьей:

засылка в триггер признака для условной передачи управления ω единицы или нуля в зависимости от знака числа в сумматоре;

выработка признака переполнения φ;

переход к строке a, если отсутствует переполнение ($'\varphi = 0$);

пропуск одной команды, если тумблеры T_1 и T_2 выключены ($'T_1 = 0$, $'T_2 = 0$);

переход к строке a, если тумблер T_1 включен ($'T_1 = 1$), или останов, если T_1 выключен ($'T_1 = 0$);

подготовка к выполнению следующей по номеру команды.

Действия этих строк производятся и при выполнении всех других команд данной группы. То же самое можно сказать и о разряде ξ_1 (в дальнейшем при описании операций это оговаривать не будем).

2. $K_0 = 02$ — засылка на сумматор:

$$'('K_1 \ominus '\xi_0 \cdot '\mu) \Rightarrow S; \quad 'S \Rightarrow r.$$

Остальные действия такие, как и при $'K_0 = 01$.

3. $'K_0 = 03$ — вычитание:

$$'S \ominus '('K_1 \ominus '\xi_0 \cdot '\mu) \Rightarrow S$$
$$\operatorname{sign} 'S \vee |'('K_1 \ominus '\xi_0 \cdot '\mu)| \Rightarrow r.$$

Остальные действия такие, как и при $'K_0 = 01$.

4. $'K_0 = 04$ — операция вычитания модулей:

$$|'S| - |'('K_1 \ominus 'ξ_0 \cdot 'ц)| \Rightarrow S$$
$$\operatorname{sign}'S \vee |'('K_1 \ominus 'ξ_0 \cdot 'ц)| \Rightarrow r.$$

5. $'K_0 = 05$ — умножение с накоплением в сумматоре:

$$'r \times '('K_1 \ominus 'ξ_0 \cdot 'ц) + 'S \Rightarrow S; \quad 0 \Rightarrow r.$$

Содержимое ячейки умножается на содержимое регистра АУ r, и результат прибавляется к содержимому сумматора. После этого в регистр r засылается нуль.

Остальные действия такие, как и при $'K_0 = 01$.

6. $'K = 06$ — умножение:

$$'S \times '('K_1 \ominus 'ξ_0 \cdot 'ц) \Rightarrow S, \quad 0 \Rightarrow r.$$

Остальные действия такие, как и при $'K_0 = 01$.

7. $'K_0 = 07$ — то же, что и при $'K_0 = 06$, только выполняемая операция является делением:

$$'S : '('K_1 \ominus 'ξ_0 \cdot 'ц) \Rightarrow S; \quad 0 \Rightarrow r.$$

8. $'K_0 = 26$ — операция циклического сложения содержимого сумматора с содержимым ячейки $'K_1 \ominus 'ц'ξ_0$ (с переносом из старшего в младший разряд):

$$'('K_1 \ominus 'ξ_0 \cdot 'ц) \rightarrow r.$$

Роль разрядов $ξ_0$ и $ξ_1$ та же, что и в предыдущих операциях.

Логические операции

9. $'K_0 = 10$ — присвоение знака числа содержимому сумматора:

$$|'S| \vee \operatorname{sign}'('K_1 \ominus 'ξ_0 \cdot 'ц) \Rightarrow S; \quad 'S \Rightarrow r$$
$$\boldsymbol{P}\{\operatorname{sign}'S = 1\}\, 1 \Rightarrow ω \downarrow 0 \Rightarrow ω$$
$$'C \oplus 1 \Rightarrow C$$
$$'C \Rightarrow K.$$

10. $'K_0 = 11$ — сдвиг содержимого регистра r (включая знаковый разряд) на число разрядов, равное модулю числа, расположенного в сумматоре в разрядах с 19 по

24, влево, если $\operatorname{sign} {}'S = 0$, и вправо, если $\operatorname{sign} {}'S = 1$. Результат сдвига помещается в сумматор:

$$0 \Rightarrow r$$
$$\boldsymbol{P}\{{}'S = 0\}\, 1 \Rightarrow \omega \downarrow 0 \Rightarrow \omega$$
$${}'C \oplus 1 \Rightarrow C$$
$${}^2C \Rightarrow K.$$

Операция не зависит от ${}'\xi_0$, ${}'\xi_1$, ${}'K_1$.

11. ${}'K_1 = 12$ — операция поразрядного логического умножения:

$${}'S \wedge {}'({}'K_1\, \theta\, {}'\xi_0 \cdot {}'\eta) \Rightarrow S; \quad {}'S \Rightarrow r$$
$$\boldsymbol{P}\{{}'S = 0\}\, 1 \Rightarrow \omega \downarrow 0 \Rightarrow \omega$$
$${}'C \oplus 1 \Rightarrow C$$
$${}^2C \Rightarrow K.$$

При этом ${}'K_1$ — число четное.

12. ${}'K_0 = 13$ — то же, что и при ${}'K_0 = 12$, только выполняемая операция является операцией поразрядного логического сложения.

13. ${}'K_0 = 14$ — поразрядная логическая операция «неравнозначно $\cong$»:

$${}'S \cong {}'({}'K_1 \ominus {}'\xi_0 \cdot {}'\eta) \Rightarrow S; \quad {}'S \Rightarrow r$$
$$\boldsymbol{P}\{{}'S = 0\}\, 0 \Rightarrow \omega \downarrow 1 \Rightarrow \omega$$
$${}'C \oplus 1 \Rightarrow C$$
$${}^2C \Rightarrow K.$$

14. ${}'K_0 = 15$ — операция нормализации: содержимое сумматора сдвигается влево на число разрядов, равное числу нулей между запятой и первой значащей цифрой, если $|{}'S| < \frac{1}{2}$, и вправо на один разряд, если $|{}'S| > 1$. После сдвига число записывается по адресу ${}'K_1 \ominus {}'\xi_0 \cdot {}'\eta$, а в сумматоре фиксируется порядок числа — число, соответствующее количеству сдвигов (в первом случае — отрицательное, во втором — положительное). Знак порядка в сумматоре записывается в знаковом разряде; младшим разрядом является 19-й разряд сумматора; и далее:

$$\boldsymbol{P}\{{}'({}'K_1 \ominus {}'\xi_0 \cdot {}'\eta) = 0\}\, 1 \Rightarrow \omega \downarrow 0 \Rightarrow \omega$$
$${}'C \oplus 1 \Rightarrow C$$
$${}^2C \Rightarrow K.$$

15. $'K_0 = 16$ — посылка из сумматора по адресу:

$$'S \Rightarrow 'K_1 \ominus '\xi_0 \cdot 'ц$$
$$\boldsymbol{P}\{\text{sign}\,'S = 1\}\,1 \Rightarrow \omega \downarrow 0 \Rightarrow \omega$$
$$'C \oplus 1 \Rightarrow C$$
$$^2C \Rightarrow K.$$

16. $'K_0 = 17$ — посылка на регистр АУ:

$$'('K_1 \ominus '\xi_0 \cdot 'ц) \Rightarrow r$$
$$\boldsymbol{P}\{'r = 0\}\,1 \Rightarrow \omega \downarrow 0 \Rightarrow \omega$$
$$'C \oplus 1 \Rightarrow C$$
$$^2C \Rightarrow K.$$

17. $'K_0 = 20$ — посылка на сумматор:

$$a \ldots {}'K_1 \ominus '\xi_0 \cdot 'ц \Rightarrow S; \quad 'S \Rightarrow r.$$

Число $'K_1 \ominus '\xi_0 \cdot 'ц$ (а не содержимое этого адреса, как в операции 02) заносится в сумматор с 19 по 30 разряды. Знаком числа $'K_1 \ominus '\xi_0 \cdot 'ц$ считается содержимое ξ_1, и $'\xi_1$ посылается в знаковый разряд сумматора:

$$\boldsymbol{P}\{\text{sign}\,'S = 1\}\,1 \Rightarrow \omega \downarrow 0 \Rightarrow \omega$$
$$'C \oplus 1 \Rightarrow C$$
$$^2C \Rightarrow K.$$

Операции передачи управления

Операции передачи управления не меняют содержимого адресов машины и касаются лишь содержимого некоторых специальных регистров.

18. $'K_0 = 21$ — условная передача управления:

$$\boldsymbol{P}\{'\omega = 1\}\,'K_1 \ominus '\xi_0 \cdot 'ц \Rightarrow C \downarrow 'C \oplus 1 \Rightarrow C$$
$$^2C \Rightarrow K.$$

Если в триггере ω находится 1, то совершается переход к команде, хранящейся по адресу $'K_1 \ominus '\xi_0 \cdot 'ц$, в противном случае — к следующей по номеру команде.

19. $'K_0 = 22$ — безусловная передача управления:

$$'K_1 \ominus '\xi_0 \cdot 'ц \Rightarrow C$$
$$^2C \Rightarrow K.$$

20. $'K_0 = 23$ — операция передачи управления по ключу. Если ключ χ_i, номер которого i содержится в регистре K_1, включен, пропускается одна команда, если он выключен,— совершается переход к следующей команде, т. е.

$$\boldsymbol{P}\{'\chi_{'K_1} = 1\}\,'C \oplus 2 \Rightarrow C \downarrow 'C \oplus 1 \Rightarrow C$$
$$^2C \Rightarrow K.$$

21. $'K_0 = 25$ — начало групповой операции. По этой операции подготовляется содержимое регистра циклов $ц$:

$$'K_1 \Rightarrow ц$$
$$'C \oplus 1 \Rightarrow C$$
$$^2C \Rightarrow K.$$

22. $'K_0 = 24$ — конец групповой операции:

$$\boldsymbol{P}\{'ц \neq 0\}\,'K_1 \ominus {}'\xi_0 \cdot 'ц \Rightarrow C,\quad 'ц \ominus {}'\xi_1 \ominus 1 \Rightarrow ц \downarrow 'C \oplus 1 \Rightarrow C$$
$$^2C \Rightarrow K.$$

Первая строка означает передачу управления циклу (команде, хранящейся по адресу $'K_1 \ominus {}'\xi_0 \cdot 'ц$), если $'ц \neq 0$, и выход из цикла в противном случае, а также — уменьшение содержимого регистра циклов $ц$ на единицу, если $'\xi_1 = 0$ (для групповой операции по коротким ячейкам), и на два, если $'\xi_1 = 1$ (для групповой операции по полным ячейкам).

23. $'K_0 = 30$ — операция изменения команд. По этой команде:

$$'K_1 \ominus {}'\xi^0 \cdot 'ц \Rightarrow K$$
$$'C \oplus 1 \Rightarrow C$$
$$'K \oplus {}^2C \Rightarrow K.$$

Таким образом, на регистр команд поступает следующая команда, измененная на величину содержимого адреса $'K_1 \ominus {}'\xi_0 \cdot 'ц$.

24. $'K_0 = 37$ — по этой команде машина останавливается и, кроме того, на сумматор засылается содержимое ячейки $'K_1 \ominus {}'\xi_0 \cdot 'ц$:

$$'('K_1 \ominus {}'\xi_0 \cdot 'ц) \Rightarrow S;!$$

25. $'K_0 = 31$ — подготовительная команда для операции передачи кодов; $'K_1$ — адрес первой ячейки внутреннего ЗУ (магнитного барабана), с которой начинается операция. Операции обращения к внешним устройствам кодируются последовательностью в три команды:

$$\begin{array}{cc} 31 & \alpha \\ \beta_1 & \beta \\ 00 & \gamma. \end{array}$$

Здесь 31, β_1, 00 — коды, соответствующие регистру K_0; α, β, γ — коды, соответствующие регистру K_1. При этом для β_1 допустимы следующие значения: 01; 02; 03, которые и определяют вид операции; β — номер зоны магнитной и перфорированной лент.

Если:

1) $\beta_1 = 01$, выполняется операция передачи кодов из вводного устройства (перфорированная лента) во внутреннее ЗУ (магнитный барабан);

2) $\beta_1 = 02$, выполняется операция передачи кодов из внешнего (магнитная лента) во внутреннее ЗУ;

3) $\beta_1 = 03$, выполняется операция передачи кодов из внутреннего во внешнее ЗУ.

В каждом из этих случаев код γ определяет адрес последней ячейки внутреннего ЗУ, до которой распространяется операция. После операции передачи кодов выполняется следующая по номеру команда.

26. $'K_0 = 32$ — печать содержимого сумматора. Печатаются на бумажную ленту или перфоленту восьмеричные (вывод программы) или десятичные (вывод числового материала) коды в зависимости от положения соответствующих тумблеров на печатающем устройстве и ПУ.

27. $'K_0 = 34$ — пропуск одной строки на бумажной ленте.

5. СПЕЦИФИКА ВЫПОЛНЕНИЯ НЕКОТОРЫХ ОПЕРАЦИЙ И ПРИМЕРЫ ИХ ИСПОЛЬЗОВАНИЯ

Операция 01. После выполнения этой операции результат ее в обратном коде находится в сумматоре АУ ($'S$ = сумме), а второе слагаемое со знаком суммы — в регистре АУ. Если в результате выполнения этой операции

получается нуль, то он всегда имеет отрицательный знак (за исключением случая $+0+(+0)=+0$), т. е. при сложении чисел, равных по абсолютной величине, но противоположных по знаку, $'S=-0$ и вырабатывается сигнал $'\varphi=1$. Это можно использовать для построения счетчиков. Если, например, необходимо повторить некоторую группу команд n раз, то перед ней следует поставить команды засылки числа $-(n-1)$ в некоторую ячейку для счетчика α:

```
20  4000 ⊕ (n ⊖ 1)
16  α,
```

а после этой группы счетчик вида

```
20  0001
01  α
16  α
21  (начальный адрес повторяющейся группы команд).
```

Команда условной передачи управления, стоящая в конце счетчика, выполняется столько раз, сколько раз вырабатывается сигнал $'\varphi=1$ после команды 01 α. Так как первоначально $'\alpha=-(n-1)$, а затем $|'\alpha|$ уменьшается на единицу, оставаясь отрицательным, то это число равно n.

Операция 02. После выполнения этой операции $'S='r$.

Операция 03. После этой операции $'S$ равно результату операции, $'r$ — второму слагаемому со знаком результата. Если $'S$ равно нулю, то нуль всегда отрицательный, за исключением случая $+0-(-0)=+0$. Операция 03 также может быть использована при построении счетчиков.

Операция 04. После этой операции $'S$ равно результату, $'r$ — второму слагаемому со знаком результата. Нуль результатов всегда отрицательный. С помощью этой операции удобно реализовать предикатную формулу

$$\boldsymbol{P}\{|a|\leqslant|b|\}\alpha\downarrow\beta.$$

Если $'r_1=a$, $'r_2=b$, то этой формуле соответствуют команды:

```
02  r1
04  r2
21  α
22  β.
```

Операция 04 может быть использована для присвоения величине знака + или — независимо от ее собственного.

Присвоение $'S$ (содержимому сумматора) знака +. Пусть $'\alpha = 0$. Тогда модуль содержимого сумматора можно получить с помощью одной команды

$$04\ \alpha \quad |'S| - |'\alpha| = |'S| \Rightarrow S.$$

Присвоение знака — содержимому ячейки a:

20	0000	$0 \Rightarrow S$
04	a	$\|'S\| - \|'a\| = \|'a\| \Rightarrow S$
16	a	$'S \Rightarrow a.$

Операция 05. После выполнения операции 05 в S находится результат операции, а $'r = 0$. Чаще всего операция 05 употребляется после операции 17.

Так, последовательность команд

02	a
17	b
05	c

реализует адресную формулу

$$'a + 'b \times 'c \Rightarrow S.$$

Последовательность команд

17	b
05	c

осуществляет адресную программу

$$'b \Rightarrow r$$
$$'S + 'r \cdot 'c = 'S + 'b \cdot 'c \Rightarrow S.$$

Однако операция 05 может следовать и за другими операциями:

1. За операцией 02:

а) 02 a $\left.\begin{array}{l} 'a \Rightarrow r; \quad 'a \Rightarrow S \\ 05\ b \end{array}\right\} 'S + 'r \cdot 'b = 'a + 'a \cdot 'b = 'a \cdot (1 + 'b) \Rightarrow S;$ $0 \Rightarrow r$

б) 02 a $\left.\begin{array}{l} 'a \Rightarrow r; \quad 'a \Rightarrow S \\ 05\ a \end{array}\right\} 'S + 'r \cdot 'a = 'a + ('a)^2 = 'a \cdot (1 + 'a) \Rightarrow S;\ 0 \Rightarrow r.$

2. За операцией 01:

а) 01 a) $'S + 'a \Rightarrow S$; $(\operatorname{sign} 'S) \cdot |'a| \Rightarrow r$
05 b) $'S + 'r \cdot 'b = 'S + (\operatorname{sign} 'S \cdot \operatorname{sign} 'b) \cdot |'a| \cdot |'b| \Rightarrow S$

б) 02 a) $'a \Rightarrow S$; $'a = r$
01 a) $'S + 'a = 2 \cdot 'a \Rightarrow S$; $'a \Rightarrow r$
05 b) $'S + 'r \cdot 'b = 2 \cdot 'a + 'a \cdot 'b = 'a \cdot (2 + 'b) \Rightarrow S$;
$0 \Rightarrow r$.

3. За операцией 03:

03 a) $'S - 'a \Rightarrow S$; $(\operatorname{sign} 'S) \cdot |'a| \Rightarrow r$
05 b) $'S + 'r \cdot 'b = 'S + (\operatorname{sign} 'S \cdot \operatorname{sign} 'b) \cdot |'a| \cdot |'b| \Rightarrow S$;
$0 \Rightarrow r$.

4. За операцией 04:

04 a) $|'S| - |'a| \Rightarrow S$; $(\operatorname{sign} 'S) \cdot |'a| \Rightarrow r$
05 b) $'S + 'r \cdot 'b = 'S + (\operatorname{sign} 'S \cdot \operatorname{sign} 'a) \cdot |'a| \cdot |'b| \Rightarrow S$;
$0 \Rightarrow r$.

Операция 05 применима для вычисления суммы парных произведений или суммы квадратов.

Пример. Вычисление $\sum_{i=1}^{n} a_i^2$, где $'(\alpha + i) = a_i$; $i = 0, 1, \ldots, n-1$, может быть выполнено программой

```
    20  0000
    25  n — 1
→—17  α + (n — 1)
|   —05  α + (n — 1)
|——24.
```

Операция 06. После выполнения операции 06: $'S$ — результат умножения, $'r = 0$. При умножении $'S \neq 0$ на $+0$ или -0 результат умножения всегда равен $+0$. При умножении нуля на число, отличное от нуля, правило знаков сохраняется, т. е.

$$\pm a\,(\pm 0) = +0;$$
$$(+0)\,(+a) = (-0)\,(-a) = +0;$$
$$(+0)\,(-a) = (-0)\,(+a) = -0.$$

Эта операция может быть использована:

а) для получения произведения $\prod_{i=1}^{n} '(a \oplus i)$ (без допол-

нительной пересылки содержимого сумматора в ячейки ПМБ). Адресной формуле

$$\prod_{i=1}^{n} {}'(a \oplus i) \Rightarrow S$$

соответствует программа машины «Урал»

```
    02 a ⊕ 1
    25 n ⊖ 2
|→—06 a ⊕ n
|___24;
```

б) для арифметического сдвига, т. е. для сдвига кода вправо на n разрядов с сохранением знакового разряда. Это реализуется по команде 06α, где ${}'\alpha = 2^{-n}$. Однако в некоторых случаях в результате округления последнего разряда в сдвинутом коде может измениться чередование нулей и единиц. Вследствие округления результата умножения многократное умножение любого неравного нулю числа на 1/2 не дает машинного нуля. Например, $0{,}00\ldots001 \times 0{,}1 = 0{,}00\ldots001$.

При помощи операции 06 можно содержимое сумматора уменьшить в 10^k раз:

$$06\,a,$$

где ${}'a = 10^{-k}$.

Операция 07. После выполнения деления ${}'S$ равно частному, ${}'r = 0$. При делении ± 0 на число $\neq 0$ всегда получается $+0$:

$$\frac{\pm 0}{\pm a} = +0.$$

При делении нуля на нуль с одинаковыми знаками получается результат, равный $+2-2^{-35}$, а с разными знаками он равен $+0$, т. е.

$$\frac{+0}{+0} = \frac{-0}{-0} = 2-2^{-35}; \frac{+0}{-0} = \frac{-0}{+0} = +0.$$

С помощью операции 07 можно ${}'S$ умножать на 10^k:

$$07\alpha,$$

где ${}'\alpha = 10^{-k}$.

Операция 10. После выполнения операции ${}'S = {}'r$.

Операция 11. Перед выполнением этой операции необходимо, чтобы в сумматоре содержалась константа сдвига,

расположенная в разрядах с 19 по 24, а ее знак — в знаковом разряде.

Операция 11 следует непосредственно за операцией 17. Как известно, операция 11 является логическим сдвигом, но она может быть использована как сдвиг арифметический (с сохранением знакового разряда). Для этого предварительно определяется модуль сдвигаемого числа, затем он сдвигается и результату присваивается прежний знак числа.

Пример. Сдвинуть содержимое ячейки a на 3 разряда влево с сохранением знака (умножить $'a$ на 2^3 без округления). Пусть $'\alpha = 0$, $'a$ — сдвигаемое число. Имеем программу

```
02 a  'a → S
04 α  |'S| — |'a| = |'a| → S
16 ω  |'a| → ω
20 0003 3 → S
17  ω  'ω → r
11        ('r сдвигается влево на 3 разряда и засылается в S)
10  a     (sign 'a) · |'S| → S
16  ω     'S → ω.
```

С помощью операции 11 удобно расшифровывать шкалу, построенную из 0 или 1. Если заранее составлена шкала, т. е. код, в разрядах которого единицы означают переход по программе в одном направлении, а нули в другом, то для распознавания содержимого разрядов шкалы используется бегающая единичка, первоначально засланная в один из фиксированных разрядов.

Пример:

$'\alpha = 0{,}011011 \ldots 011$ — шкала;
$'\beta = 0{,}000000 \ldots 001$ — бегающая единичка.

Программа расшифровки шкалы:

```
20 0001
17 β
11
21 k3 (выход в случае окончания просмотра всей шкалы)
16 β
12 α
21 k1 (выход в случае нуля в данном разряде шкалы)
22 k2 (выход в случае единицы в разряде шкалы).
```

Операции 12, 13 и 14. После выполнения этих операций содержимые сумматора и регистра r совпадают, т. е. $'S = 'r$. Обычно адрес при таких операциях должен

быть четным. Если при этих операциях стоит нечетный адрес, то в операции участвует его содержимое, поступающее в младшую половину сумматора.

Пример. Пусть адрес a — нечетный и $'a = 777777$, $'S = 377777\ 703707$. По команде 12 a выделяется младшая половина сумматора, т. е. после выполнения операции имеем

$$'S = 000\ 000\ 703\ 707.$$

Аналогично и при выполнении операций 13 и 14 с нечетными адресами. Если же при этих операциях стоит нечетный адрес с признаком полноты, то такой признак не принимается во внимание, т. е. команда $\theta\ 4000 \oplus a$ (где a — нечетный адрес, $\theta = 12,\ 13,\ 14$) равносильная команде $\theta\ a$.

Операция 15. Эта операция применяется при программировании задач с плавающей запятой. Так как машина «Урал» работает с фиксированной запятой, то с помощью операции 15 можно реализовать плавающую запятую программно. Если в адресной части команды с этой операцией стоит четная ячейка с признаком полноты или любая ячейка без этого признака, то в данную ячейку переписывается мантисса или половина мантиссы старшей половины содержимого сумматора. Если же стоит нечетный адрес с признаком полноты, то в соответствующую нечетную короткую ячейку запишется часть мантиссы со второй половины сумматора.

Операцию 15 можно применять для проверки $\boldsymbol{P}\{'S = 0\}$:

15 α
21 β выход в случае $'S = 0$.
22 γ выход в случае $'S \neq 0$.

Операция 16. После выполнения этой операции $'S$ и $'r$ не меняются. Как уже отмечалось, между разрядными сетками сумматора и полной ячейки машинного барабана имеется некоторое несоответствие.

Разрядные сетки дробной части сумматора и ячейки барабана полностью совпадают. Знаковый разряд сумматора — 37-ой, ячейки — 36-ой. В 36-ом разряде сумматора хранится целая часть числа, для которой на магнитном барабане нет соответствующего разряда. При выполнении операции 16 содержимое разрядов 1—36 сумматора переписывается в соответствующие 36 разрядов ячейки. Если $|'S| < 1$, то содержимое 37-го разряда равно содержимо-

му 36-го разряда, т. е. в 36-ом разряде сумматора стоит знак числа, который переписывается соответственно в знаковый разряд ячейки. Если $|'S| \geqslant 1$, то в 36-ом разряде сумматора находится целая часть числа, которая по операции 16 записывается в 36-й (знаковый) разряд ячейки. Таким образом, если в S имеется положительное переполнение, то в ячейку запишется дробная часть $'S$ со знаком —.

Пример:

$$'S = 01{,}1011\ \ldots\ 01 = +\,1{,}1011\ \ldots\ 01.$$

После операции 16 a

$$'a = 1{,}1011\ \ldots\ 01,\ \text{что означает} -0{,}1011\ \ldots\ 01.$$

Выполнив команду 02 a, получим

$$'S = 11{,}0100\ \ldots\ 10.$$

Если в S отрицательное переполнение, то в ячейку запишется дробная часть $'S$ со знаком $+$.

Пример:

$$'S = 10{,}0100\ \ldots\ 10 = -\,1{,}1011\ \ldots\ 01.$$

После команды 16 a

$$'a = 0{,}1011\ \ldots\ 01 = +\,1{,}1011\ \ldots\ 01.$$

Выполнив команду 02 a, получим

$$'S = 00{,}1011\ \ldots\ 01.$$

Операцией 16 можно посылать на магнитный барабан содержимое всего сумматора, содержимое старшей половины сумматора, а также содержимое младшей половины сумматора.

Если при операции 16 стоит четный адрес с признаком полноты, то весь сумматор переписывается в полную ячейку; для адреса без признака полноты (адрес короткой ячейки) в нее записывается старшая половина сумматора. Если при операции 16 стоит нечетный адрес с признаком полноты, то в соответствующую нечетную короткую ячейку записывается младшая половина содержимого сумматора.

Пример. Пусть код числа $'S$ равен 10101 ... 11.

После выполнения команды 16 4015 в ячейку 0015 запишется код 110100 ... 011, стоящий во второй половине сумматора.

Операция 17. После этой операции обычно ставятся операции 05, 11. Если после нее стоит операция 21, то по последовательности операций проверяется содержимое ячейки на равенство нулю:

17 a
21 α выход в случае $'a = 0$
22 β выход в случае $'a \neq 0$.

Операция 20. После выполнения этой операции

$$'S = 'r.$$

Операции 25 и 24. Группу команд, которую необходимо многократно выполнять, заключают между операциями 25 и 24, указывая при операции 25 в адресной части команды количество циклов (сколько раз эта группа команд должна исполняться). Если перед любой командой стоит минус (признак изменяемости адреса в цикле), то в этой команде адрес уменьшается на $'ц$ и выполняется измененная команда. Если в команде 25 n $n \geqslant 4000$, то $ц$ уменьшается на 2, если $n < 4000$, то $ц$ уменьшается на 1, т. е. говорят, что цикл работает по полным или по неполным ячейкам, или с шагом 2 или с шагом 1. Можно осуществить циклы с шагом, отличным от 2 или от 1. Если шаг δ небольшой, то необходимо, чтобы операция 24 выполнилась подряд:

δ раз, в случае $n < 4000$:
$\frac{\delta}{2}$ раз, в случае $n \geqslant 4000$.

Пример 1. Необходимо просуммировать содержимое ячеек через 0003:

$$\alpha + 0000, \quad \alpha + 0003, \quad \alpha + 0006, \ \ldots, \quad \alpha + 3n.$$

Для этого организуем цикл по неполным ячейкам с шагом $\delta = 3$:

k	20 000	$0 \to S$
$k \oplus 1$	25 $3n$	$3n \to ц$
$k \oplus 2$	—26 $\alpha \oplus 3n$	$'S + '(\alpha + 3n - 'ц) \to S$
$k \oplus 3$	24 $k \oplus 4$	$'ц - 3 \to ц$
$k \oplus 4$	24 $k \oplus 5$	$P\{('ц > +0)\ k+2$
$k \oplus 5$	24 $k \oplus 2$	

Если шаг цикла большой, то его можно осуществить несколько иначе:

а) по неполным ячейкам

```
k          25 n1                    n1 = amax − amin, где
k ⊕ 1   −20 4000 ⊕ n1 ⊕ δ ⊖ 1       amax, amin — максимальный и
k ⊕ 2      01 k                     минимальный из изменяющихся
k ⊕ 3      16 α                     адресов
            .
            .      } группа команд цикла; команды с изменяющимися
            .      } в цикле адресами записываются в виде — 0amax.
α          00 0000
α ⊕ 1      24 k ⊕ 1
```

Пример 2. Необходимо переслать информацию из ячеек 0511, 0531, 0551, 0571 в ячейки 0510, 0530, 0550, 0570:

$$n_1 = 0571 - 0511 = 0060;$$
$$\delta = 0531 - 0511 = 0020; \quad \delta - 1 = 0017.$$

Программа:

```
k          25 0060
k ⊕ 1   −20 4077   (4077 = 4000 + 0060 + 0017)
k ⊕ 2      01 k
k ⊕ 3      16 k ⊕ 6
k ⊕ 4   −02 0571
k ⊕ 5   −16 0570
k ⊕ 6      00 0000
k ⊕ 7      24 k ⊕ 1;
```

б) по полным ячейкам

```
k          25 n1                    n1 = amax четное − amin четное
k ⊕ 1   −20   4000 ⊕ n1 ⊕ (δ ⊖ 2)
k ⊕ 2      01 k
k ⊕ 3      16 α
            .
            .
            .
α          00 0000
α ⊕ 1      24 k ⊕ 1
```

Можно реализовать циклы с шагом 2, в которых могут быть как полные, так и неполные изменяющиеся адреса команд. При этом, если в изменяющейся команде адрес полный, то данная команда пишется со знаком — (с признаком изменения в цикле). Если же в изменяющейся команде адрес, который должен изменяться на 1, неполный, то перед этой командой ставится команда 30 *a*.

Такой цикл называется циклом по полным и неполным ячейкам одновременно:

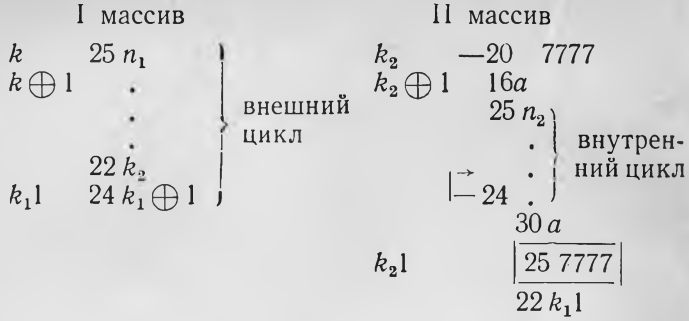
```
k          25 4000 ⊕ n
k ⊕ 1  —20 n
          06 β          'β = 1/2
          16 a
           .
           .
           .
       —θ   b_max       b_max — максимальный полный изменяющийся в цикле адрес
           .
           .
           .
          30 a
          θ c_min       c_min — минимальный короткий изменяющийся адрес
           .
           .
           .
          24 k ⊕ 1.
```

Можно также осуществить цикл внутри другого цикла:

а) в одном массиве ячеек

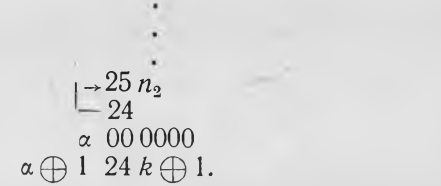
```
k          25 n_1
k ⊕ 1  —20 4000 ⊕ n_1   } группа команд внешнего
          01 k          } цикла
          16 α
           .
           .
           .
          25 n_2        }
       |→ 24             } внутренний цикл
          00 0000       }
           .            }
           .            } группа команд внешнего
           .            } цикла;
          24 k ⊕ 1      }
```

б) в двух независимых массивах ячеек



Если внешний цикл — по полным ячейкам, то $'k_2 1 = 25\,7777$, по неполным — $'k_2 1 = 25\ 3777$;

в) и в таком варианте:



Можно осуществлять циклы любой глубины в зависимости от потребности конкретной задачи. Если необходимо выйти из цикла не после его окончания, а по выполнению какого-либо другого условия, но при этом количество повторений группы команд заранее не определено, то эту группу команд необходимо заключить между командами $25n$ и $24k$, где n должно быть больше 4000 и нечетное. При этом содержимое счетчика циклов никогда не станет равным 0, а значит, и окончания цикла по счетчику не будет.

Часто необходимо, чтобы в вычислительном процессе рабочие адреса (получающиеся после изменения адреса по признаку изменяемости, т. е. те адреса, содержимое которых в данном такте работы машины участвует в

вычислении) изменялись от минимального своего значения до максимального и наоборот в одном и том же цикле.

Пример. Необходимо вычислить сумму

$$s = \sum_{i=0}^{n} a_i a_{n-i}.$$

В обычном цикле, если перед командой стоит минус (признак изменяемости адреса), рабочий адрес данной команды будет увеличиваться от минимального своего значения до максимального. Те же команды, у которых рабочие адреса должны изменяться в обратном порядке, необходимо заключить между командами вида — 25 n, т. е. с признаком изменяемости (со знаком минус).

Пусть $'(\alpha \oplus i) = a_i$ $(i = 0, \ldots, n)$ тогда программа определения s будет такой:

$0 \to S$	k 20 0000
$n \to ц$	$k \oplus 1$ 25 n
$'(\alpha \oplus n \ominus 'ц) = '(\alpha \oplus i) = a_i \to r$	$k \oplus 2$ —17 $\alpha \oplus n$
$n \ominus 'ц = i \to ц$	$k \oplus 3$ —25 n
$'(\alpha \oplus n \ominus 'ц) \cdot 'r + 'S = '(\alpha \oplus n \ominus i) \cdot 'r +$ $+ 'S = a_{n-i} a_i + 'S \to S$	$k \oplus 4$ — 05 $\alpha \oplus n$
$n \ominus 'ц = n \ominus i \to ц$	$k \oplus 5$ —25 n
$'ц \ominus 1 = n (i + 1) \to ц$	$k \oplus 6$ 24 $k \oplus 2$
$k \oplus 2 \to c$	

Операция 26. Эта операция в основном применяется для определения контрольной суммы содержимого некоторого массива ячеек. После выполнения этой операции в сумматоре S может получиться переполнение, но сигнал φ при этом схемно заблокирован, и машина на него не реагирует. Для сравнения $'S$ с предыдущей контрольной суммой после цикла контрольного суммирования необходимо осуществить команду засылки и выборки контрольной суммы в S, а затем сравнение:

20 0000
25 n
—26 $\alpha \oplus n$
24
16 α
02 α
14 β.

В некоторых случаях, когда программа с фиксированной запятой, а переполнение не влияет на вычисление, операция 26 может применяться вместо операции 01.

Операция 30. Применяется для изменений команд, стоящих непосредственно за этой операцией. После выполнения операции 30 в регистре команд будет стоять модифицированная команда, получающаяся в результате прибавления к следующей за ней команде содержимого ячейки a.

Пример. Пусть на магнитном барабане записаны команды

0300	30	0100
0301	03	4170,

а в ячейке 0100 — константа 000 402.

Содержимое счетчика команд и регистра команд при выполнении этих двух команд следующее:

счетчик команд	регистр команд
0300	30 0100
0301	03 4572

Операция 30 может использоваться многократно.

Пример:

$$\begin{array}{ll} k & 30\,\alpha \\ k \oplus 1 & 30\,\alpha_1 \\ & \cdot \\ & \cdot \\ & \cdot \\ k \oplus n \ominus 1 & 30\,\alpha_n \\ k \oplus n & 0\,a. \end{array}$$

После n-кратного выполнения операции 30 будет выполняться команда

$$'(\ldots\ '('('\alpha_1 \oplus \alpha_2) \oplus \alpha_3) \oplus \ \ldots\ \alpha_n) \oplus\ '(k \oplus n).$$

Операцию 30 можно также использовать в цикле с признаком изменяемости адреса, т. е. со знаком —

$$\begin{array}{ll} k & -30\,a \\ k \oplus 1 & \ldots \end{array}$$

После выполнения команды — 30 a будет выполняться команда

$$'(a \ominus\ 'ц) \oplus\ '(k \oplus 1).$$

Операция 31. Команда с этим кодом записывается подряд в три короткие ячейки:

$$31\,a_1$$
$$\theta\;\;c$$
$$00\,a_2,$$

где $\theta = (01,\ 02,\ 03)$.

В зависимости от четности a_1 и от значения θ команда с кодом 31 может иметь несколько разновидностей:

а) если $\theta = 01$ и a_1 — адрес без признака полноты, т. е. $a_1 < 4000$, то она будет командой ввода на МБ с перфоленты восьмеричных чисел и команд:

б) если $\theta = 01$, $a_1 \geqslant 4000$ и четно, то она является командой ввода на МБ с перфоленты десятичных чисел по полным ячейкам;

в) если $\theta = 02$, $a_1 \geqslant 4000$ и четно, то она является командой переписи с магнитной ленты на магнитный барабан по полным ячейкам;

г) если $\theta = 03$, $a_1 \geqslant 4000$ и четно, то она является командой переписи с магнитного барабана на магнитную ленту по полным ячейкам.

Адрес a_2 всегда должен быть одинаковой четности с адресом a_1, а также без признака полноты, чтобы не было переполнения при окончании ввода десятичных чисел.

В пп. б, в и г адрес a_1 (а значит, и a_2) иногда может быть нечетным. При этом перепись будет происходить только по неполным и нечетным ячейкам.

Команду 30 a можно ставить перед кодом 31, но внутри групповой команды — нельзя.

6. ПРЕДСТАВЛЕНИЕ АДРЕСНЫХ ФУНКЦИЙ

Рассмотрим методику построения адресных функций на машине «Урал».

Как было показано в гл. II, любой адресный алгоритм можно представить эквивалентной ему формой, в которой все входящие в запись алгоритма адресные функции имеют ранг не выше второго.

Учитывая специфику набора операций машины «Урал», задачу построения адресных функций на этой машине разобьем на две части:

а) получение адресной функции на сумматоре

$$f \Rightarrow S;$$

б) перенос содержимого сумматора по адресу, равному значению некоторой адресной функции f

$$'S \Rightarrow f.$$

Наиболее существенной особенностью реализации программ на машине «Урал» является возможность выполнения операций по адресам второго ранга[1] лишь в том случае, когда α является подрегистром K_1 регистра команд K, а именно адресным программам

$$'('\alpha \oplus a) \Rightarrow S, \quad {}^2\alpha \Rightarrow S \text{ или } 'S \Rightarrow '\alpha \oplus a; \quad 'S \Rightarrow '\alpha \qquad (6.1)$$

соответствуют следующие программы машины «Урал»:

Адресная программа	Эквивалентная ей адресная программа, реализуемая машиной «Урал»	Программа машины «Урал»
1. $'('\alpha \oplus b) \Rightarrow S$	$'\alpha \Rightarrow K_1$ $'('K_1 \oplus b) \Rightarrow S$	30 α 02 b
2. ${}^2\alpha \Rightarrow S$	$'\alpha \Rightarrow K_1$ ${}^2K_1 \Rightarrow S$	30 α 02 0000
3. $'S \Rightarrow '\alpha \oplus a$	$'\alpha \Rightarrow K_1$ $'S \Rightarrow 'K_1 \oplus a$	30 α 16 a
4. $'S \Rightarrow '\alpha$	$'\alpha \Rightarrow K_1$ $'S \Rightarrow 'K_1$	30 α 16 0000

Рассмотрим типовые примеры. Для упрощения перехода от адресных программ к программам машины «Урал» будем в дальнейшем приводить эквивалентные исходным адресные программы, реализуемые машиной.

Пусть 0 обозначает любую из операций машины «Урал»: 01, 03, 04, 06, 07, 10, 12, 13, 14.

---

[1] Не считая случая использования групповых операций.

а) Получение адресных функций на сумматоре:

Адресная программа	Эквивалентная ей адресная программа, реализуемая машиной «Урал»	Программа машины «Урал»
5. $a \to S$ (содержимое по адресу нулевого ранга, т. е. константу a заслать в S)	$a \to S$	20 a
6. $'a \to S$ (содержимое адреса заслать в S)	$'a \to S$	02 a
7. $a\theta'b \to S$	$a \to S$ $'S\theta'b \to S$	20 a θ b
8. $'b\theta a \to S$ (если θ—неперестановочная операция, например операция вычитания, необходимо воспользоваться данной программой; в противном случае следует, переставив местами аргументы, воспользоваться предыдущей программой)	$a \to S$ $'S \to \omega$ $'b \to S$ $'S\theta'\omega \to S$	20 a 16 ω 02 b θ ω
9. $a\theta^2 b \to S$	$a \to S$ $'b \to K_1$ $'S\theta^2 K_1 \to S$	20a 30b θ 0000
10. $a\theta\,'('b \oplus c) \to S$	$a \to S$ $'b \to K_1$ $'S\theta'('K_1 \oplus c) \to S$	20a 30b θc
11. $^2a\theta b \to S$ (для перестановочной операции θ компоненты следует поменять местами)	$b \to S$ $'S \to \omega$ $'a \to K_1$ $^2K_1 \to S$ $'S\theta'\omega \to S$	20b 16ω 30a 02 0000 $\theta\omega$
12. $'('a \oplus b)\,\theta c \to S$ (для перестановочной операции θ компоненты следует поменять местами)	$c \to S$ $'S \to \omega$ $'a \to K_1$ $'('K_1 \oplus b) \to S$ $'S\theta'\omega \to S$	20c 16ω 30a 02b $\theta\omega$
13. $'a\theta'b \to S$	$'a \to S$ $'S\theta'b \to S$	02a θb
14. $'a\theta^2 b \to S$	$'a \to S$ $'b \to K_1$ $'S\theta^2 K_1 \to S$	02a 30b θ 0000
15. $^2a\theta'b \to S$	$'a \to K_1$ $^2K_1 \to S$ $'S\theta'b \to S$	30a 02 0000 θb

Адресная программа	Эквивалентная ей адресная программа, реализуемая машиной «Урал»	Программа машины «Урал»
16. $'('a \oplus b)\,\theta'c \to S$	$'a \to K_1$ $'('K_1 \oplus b) \to S$ $'S\theta'c \to S$	30 a 02 b θc
17. $^2a\theta^2b \to S$	$'a \to K_1$ $^2K_1 \to S$ $'b \to K_1$ $'S\theta^2K_1 \to S$	30 a 02 0000 30 b θ 0000
18. $'('a+b)\,\theta'\,('c+d) \to S$	$'a \to K_1$ $'('K_1 \oplus b) \to S$ $'c \to K_1$ $'S\theta'('K_1 \oplus d) \to S$	30 a 02 b 30 c θ d
19. $^3\alpha \to S$ (содержимое по адресу третьего ранга α заслать в S)	$'\alpha \to K_1$ $^2K_1 \to K_1$ $^2K_1 \to S$ $'\alpha \to K_1$	30 α 30 0000 02 0000 30 α
20. $^n\alpha \to S$ (содержимое по адресу n-го ранга α заслать в S $(n > 3)$	$^2K_1 \to K_1$ $\vdots$ } $n-2$ раза $^2K_1 \to K_1$ $^2K_1 \to S$	30 0000 $\vdots$ } $n-2$ раза 30 0000 02 0000
21. $'('('\alpha \oplus a) \oplus b) \to S$	$'\alpha \to K_1$ $'('K_1 \oplus a) \to K_1$ $'('K_1 \oplus b) \to S$	30 α 30 a 02 b
22. $'('\alpha \oplus '\beta) \to S$	$'\alpha \to S$ $'S \oplus '\beta \to S$ $'S \to \omega$ $'\omega \to K_1$ $^2K_1 \to S$	02 α 01 β 16 ω 30 ω 02 0000
23. $'('\alpha \oplus '\beta \oplus a) \to S$	$'\alpha \to S$ $'S \oplus '\beta \to S$ $'S \to \omega$ $'\omega \to K_1$ $'('K_1 \oplus a) \to S$	02 α 01 β 16 ω 30 ω 02 a
24. $c + 'a \cdot 'b \to S$	$c \to S$ $'a \to r$ $'r \cdot 'b + 'S \to S$ (r — регистр АУ)	20 c 17 a 05 b
25. $'a \cdot 'b + 'c \to S$	$'c \to S$ $'a \to r$ $'r'b + 'S \to S$	02 c 17 d 05 b
26. $'a \cdot 'b + 'c \cdot 'd \to S$	I вариант II вариант $'a \to S$ $0 \to S$ $'S \cdot 'b \to S$ $'a \to r$ $'c \to r$ $'r \cdot 'b + 'S \to S$ $'r \cdot 'd + 'S \to S$ $'c \to r$ $'r \cdot 'd + 'S \to S$	I вариант II вариант 02 a 20 0000 06 b 17 a 17 c 05 b 05 d 17 c 05 d

б) перенос содержимого сумматора $'S$ по адресу f:

$$'S \Rightarrow f,$$

где f — некоторая адресная функция.

Поскольку в данном случае значение адресной функции f является адресом, набор допустимых функций f будет ограниченным (например, без специальных уточнений нельзя считать адресом значение адресной функции $\sin {}'a$). В этом случае функция либо будет заданным целым числом (адресом), либо результатом применения операций сложения, вычитания и штрих-операции. При построении функции f могут также встретиться логические операции ($\wedge$, $\vee$, ...), операция умножения, в редких случаях — деления и др.

	Адресная программа	Эквивалентная ей адресная программа, реализуемая на машине «Урал»	Программа машины «Урал»
27.	$'S \to a$	$'S \to a$	16 a
28.	$'S \to {}'a \oplus b$	$'a \to K_1$ $'S \to {}'K_1 \oplus b$	30 a 16 b
29.	$'S \to {}'a$	$'a \to K_1$ $'S \to {}'K_1$	30 a 16 0000
30.	$'S \to {}'({}'a \oplus b) \oplus c$	$'a \to K_1$ $'({}'K_1 \oplus b) \to K_1$ $'S \to {}'K_1 \oplus c$	30 a 30 b 16 c
31.	$'S \to {}'a \oplus {}'b \oplus c$	$'S \to \omega 1$ $'a \to S$ $'S + {}'b \to S$ $'S \to \omega 2$ $'\omega 1 \to S$ $'\omega 2 \to K_1$ $'S \to {}'K_1 \oplus c$	16 $\omega 1$ 02 a 01 b 16 $\omega 2$ 02 $\omega 1$ 30 $\omega 2$ 16 c
32.	$'S \to {}^2a$	$'a \to K_1$ ${}^2K_1 \to K_1$ $'S \to {}'K_1$	30 a 30 0000 16 0000
33.	$'S \to {}^n a$	$'a \to K_1$ $'K_1 \to K_1$ $\vdots$ ${}^2K_1 \to K_1$ } $n-1$ раза $'S \to K_1$	30 a 30 0000 $\vdots$ 30 0000 } $n-1$ раза 16 0000

7. ПОСТРОЕНИЕ СХЕМ ОБОЗРЕВАНИЯ

Программы, реализующие схемы обозревания последовательностей и других массивов, для машины «Урал»

могут строиться с помощью групповых операций (коды 25 и 24), операции переадресации и соответствующих программных счетчиков, а также с помощью операции изменения команд (код 30). Проиллюстрируем сказанное типовыми примерами.

Пример 1. *Вычисление суммы* $S = \sum_{i=0}^{n-1} a_i$.

Исходное отображение

$$'(\alpha \oplus id) = a_i, \ (i = 0,1, \ \ldots \ n-1). \tag{6.2}$$

Здесь при суммировании по полным ячейкам $d = 2$, α — четное, а по неполным ячейкам $d = 1$, α — произвольное.

Результат требуется получить по адресу γ.

А. Адресную программу вычисления суммы представим с помощью формулы циклирования в виде

$$\begin{array}{l} 0 \to S \\ \boldsymbol{Ц}\,\{n \ominus 1\ (-1)\ 0 \to u\} \\ 'S + '(\alpha \oplus d\ (n \ominus 1) \ominus d' \cdot u) \to S \\ 'S \to \gamma \\ \boldsymbol{Я} \end{array} \tag{6.3}$$

или без нее

$$\begin{array}{l} d \cdot (n \ominus 1) \to p \\ 0 \to S \\ R \ \ldots \ 'S + '(\alpha \oplus (n \ominus 1) \cdot d \ominus p) \to S \\ 'p \ominus d \to p \\ P\,\{'p \geqslant -0\}\ R \\ 'S \to \gamma \\ \boldsymbol{Я}. \end{array} \tag{6.4}$$

Программе (6.3) при суммировании по неполным ячейкам соответствует следующая программа машины «Урал», построенная на групповых операциях:

$$\begin{array}{lll} & 20 & 0000 \\ & 25 & n \ominus 1 \\ k \ \ldots & -01 & \alpha \oplus n \ominus 1 \\ & 24 & k \\ & 16 & \gamma \end{array} \tag{6.5}$$

а по полным:

$$\begin{array}{lll} & 20 & 0000 \\ & 25 & 4000 \oplus 2 \cdot n \ominus 2 \\ k \ \ldots & -01 & 4000 \oplus \alpha \oplus 2 \cdot n \ominus 2 \\ & 24 & k \\ & 16 & \gamma \end{array} \tag{6.6}$$

Здесь k — адрес начальной команды цикла. Знак минус перед этой командой означает признак изменяемости (единица в разряде, соответствующем регистру ξ_0).

Для превращения приведенных программ для машины «Урал» в стандартные подпрограммы необходимо прежде всего дополнить их приказами, по которым можно совершать возврат с этих подпрограмм на основную программу. Это может быть достигнуто добавлением в конце каждой из них следующих двух команд:

$$\begin{array}{ll} 30 & \omega \\ 22 & 0000 \end{array} \qquad (6.7)$$

По адресу ω в основной программе перед уходом на подпрограмму помещается адрес команды, на которую должен быть совершен возврат от подпрограммы. Команды (6.7) эквивалентны в этом случае метке $Я$ в адресных программах, если в последних переход на подпрограмму осуществляется по формуле вхождения (вторая из меток которой соответствует $'\omega$).

Программы (5.5) и (6.6) зависят от количества суммируемых членов и от места их размещения в памяти.

Б. Для устранения зависимостей от указанных параметров составим предварительно соответствующую адресную программу. Введем фиксатор конца последовательности

$$'\varphi = \alpha \oplus (n \ominus 1) \cdot d; \quad (^2\varphi = a_{n-1})$$

и пусть $'c = n \ominus 1$. Тогда получим адресную программу

$$\begin{array}{l} 0 \to S \\ \boldsymbol{Ц}\,\{'c\,(-1)\,0 \to u\} \\ 'S + '('\varphi \ominus d \cdot 'u) \to S \\ 'S \to \gamma \\ Я. \end{array} \qquad (6.8)$$

Программа в кодах машины «Урал» имеет вид

$$\begin{array}{lll} 20 & 0000 & \\ 30 & c & \\ 25 & 4000 \oplus 0000 & \\ R \ldots 30 & \varphi & \\ -01 & 4000 \oplus 0000 & \\ 24 & R & \\ 16 & \gamma & \\ 30 & \omega & \\ 22 & 0000 & \end{array} \qquad \begin{array}{l} 0 \to S \\ 'c \to K_1 \\ 'K_1 \to u \\ R \ldots {}'\varphi \to K_1 \\ 'S + '('K_1 \ominus d \cdot 'u) \to S \\ 'u \ominus d \to u \\ \boldsymbol{P}\,\{'u \leqslant -0\}\,R \\ 'S \to \gamma \\ Я. \end{array} \qquad (6.9)$$

Для сравнения справа приведена покомандная адресная запись программы для машины «Урал», эквивалентная программе (6.8).

Таким образом, для устранения зависимости от параметра l его необходимо поместить по адресу φ, а команды, содержащие этот параметр, т. е. команды вида θl, (где θ — код операции) заменить двумя командами

$$\begin{array}{ll} 30 & l \\ \theta & 0000. \end{array}$$

Это замечание не касается команд обращения к МЛ и ПЛ.

В. Та же задача вычисления значения суммы может быть решена с помощью программы со специальным программным счетчиком с операцией переадресации.

Рассмотрим случай, когда a_i размещены в коротких ячейках (сравни адресную программу (6.4)):

```
    20  40000 ⊕ n ⊖ 1 } — (n — 1) → r1 — исходное заполнение счет-
    16      r1        } чика r1
    20  0000  }
    16  γ     } 0 → γ — очистка ячейки для накапливания в ней
              } суммы
R   02  α ⊕ 0 }
    01  γ     } 'γ + '(α ⊕ 0) → γ
    16  γ     }
    20  0001  }
    01  R     } изменение команды с меткой R — увеличение на еди-
    16  R     } ницу кода, соответствующего регистру K1 (пере-
              } адресация команды R)
    20  0001  }
    01  r1    }
    16  r1    } счетчик по числу элементов в сумме.
    21  R     }
    30  ω
    22  0000
```

В данной программе роль адреса μ адресной программы (6.4) выполняет рабочая ячейка $r1$, а адреса S — ячейка γ.

Г. Приведем для сравнения программу, использующую программный счетчик, и команду модификации команд. В отличие от предыдущей получим программу, запись которой сохраняется неизменной в процессе работы программы,

```
    20  4000 ⊕ n ⊖ 1 } — (n — 1) → r
    16      r        }
    20      0000     }
    16      γ        } 0 → γ
    16      a          0 → a
R . . . 30  a          'a → K1
    02      α ⊕ 1      '('K1 ⊕ α ⊕ 1) → S
    01      γ          'S + 'γ → S
    16      γ          'S → γ
    20      0001     }
    01      a        } 'a + 1 → a
    16      a        }
    20  0001       }
    01      r      } 'r + 1 → r }
    16      r      }            } счетчик r
    21      R      }            }
    30      ω                   }
    22  0000.
```

Пример 2. *Вычисление значений многочлена*

$$f(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n =$$
$$= (\ldots(a_0x + a_1)x + \ldots)x + a_n$$

Пусть дано адресное отображение

$$'(\alpha \oplus i) = a_i \ (i = 0, 1, \ldots, n);$$
$$'c = n; \ '\varphi = \alpha \oplus n; \ '\beta = x.$$

Адресная программа

$0 \to S$
$\boldsymbol{Ц}\{'c(-1)0 \to \mu\}$
$'S \times 'x + '('\varphi \oplus '\mu) \to S$
$'S \to \gamma$
$\boldsymbol{Я}$

Программа для машины «Урал»

```
      20  0000
      30  c
      25  0000
R ... 06  β
      30  φ
     —01  0000
      24  R
      16  γ
      30  ω
      22  0000.
```

Пример 3. *Вычисление суммы первых положительных членов последовательности* $a_1, a_2, \ldots, a_n$ *(по первый отрицательный член).* Результат требуется поместить по адресу β.

Пусть $'(\alpha \oplus i) = a_i$.

А. Программы, зависящие в явном виде от параметров α и n:

Адресная программа

$0 \to S$
$\boldsymbol{Ц}\{n \ominus 1(-1)0 \to \mu\}$
$\boldsymbol{Р}\{'(\alpha \oplus n \ominus '\mu) > 0\}\ 'S + '(\alpha \oplus n \ominus '\mu) \to S$
$'S \to \beta$
$\boldsymbol{Я}$

Программа для машины «Урал»

```
      20  0000
      16  β
      25  n — 1
R2 ...—02  α ⊕ n
      21  R1
      01  β
      16  β
      24  R2
R1 ... 30  ω
      22  0000.
```

Б. Программы, не зависящие в явном виде от параметров α и n. Пусть

$$'(\alpha \oplus i) = a_i; \quad 'c = n \ominus 1; \ '\varphi = \alpha \oplus n.$$

Адресная программа

$0 \to \beta$
$\boldsymbol{Ц}\{'c(-1)0 \to \mu\}$
$\boldsymbol{Р}\{'('\varphi \ominus \mu) > 0\}\ '\beta + '('\varphi \ominus '\mu) \to \beta$
$\boldsymbol{Я}$

Программа для машины «Урал»

```
      20 0000
      16 β
      30 c
      25 0000
R2 ... 30 φ
      — 02 0000
      21 R1
      01 β
      16 β
      24 R2
R1 ... 30 ω
      22 0000.
```

Пример 4. Вычисление суммы чисел $a_1, a_2, \ldots, a_n$, расположенных по адресам $\alpha_1, \alpha_2, \ldots, \alpha_n$, не образующим закономерного ряда. Результат требуется поместить по адресу β.

Поместить адреса $\alpha_1, \alpha_2, \ldots, \alpha_n$ в последовательность адресов $\alpha \oplus 1, \alpha \oplus 2, \ldots, \alpha \oplus n$, т. е. пусть

$$'(\alpha \oplus i) = \alpha_i \text{ или } {}^2(\alpha \oplus i) = \alpha_i, \tag{6.10}$$

А. Программы, включающие в явном виде параметры α и n

Адресная программа

$$\begin{array}{c} 0 \to S \\ Ц\{n \ominus 1\ (-1)\ 0 \to ц\} \\ 'S + {}^2(\alpha \oplus n \ominus 'ц) \to S \\ 'S \to \beta \\ Я \end{array}$$

Программа для машины «Урал»

```
        20 0000
        25 n ⊖ 1
R ... —30 α ⊕ n
        01 0000
        24 R
        16 β
        30 ω
        22 0000
```

Б. Программы, не зависящие от параметров α и n. Пусть кроме (6.10) имеем $'c = n \ominus 1$, $'\varphi = \alpha \oplus n$.

Адресная программа

$$\begin{array}{c} 0 \to S \\ Ц\{'c\ (-1)\ 0 \to ц\} \\ 'S + {}^2('\varphi \ominus 'ц) \to S \\ 'S \to \beta \\ Я \end{array}$$

Программа для машины «Урал»

```
        20 0000
        30  c
        25 0000
R ...   30 φ
       —30 0000
        01 0000
        24 R
        16 β
        30 ω
        22 0000.
```

Пример 5. *Групповой перенос информации.* Элементы информации, расположенные по адресам

$$\alpha, \alpha \oplus d, \alpha \oplus 2d, \ldots, \alpha \oplus nd,$$

перенести в последовательность адресов

$$\beta, \beta \oplus p, \beta \oplus 2p, \ldots, \beta \oplus np$$

(d и p — некоторые целые числа).

Составим программы, не зависящие от α, β и n

Пусть $'c = n$; $'\varphi = \alpha$; $'\psi = \beta$.

Адресная программа

$$\begin{array}{c} Ц\,\{0\,(1)\,'c \to \pi\} \\ '('\varphi \oplus '\pi d) \to '\psi \oplus '\pi \cdot p \\ Я. \end{array}$$

Программа для машины «Урал»

```
        30 c
        25 0000
R . . . 30 φ
        02 0000
        30 ψ
        16 0000
        20 d
        01 φ
        16 φ
        20 p
        01 ψ
        16 ψ
        24 R
        30 ω
        22 0000.
```

Рекомендуем читателю составить программу, не зависящую также от параметров p и d.

Пример 6. *Групповой перенос информации из последовательности адресов*

$$\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_n$$

в последовательность адресов

$$\beta_1,\ \beta_2,\ \ldots,\ \beta_n,$$

где ни одна из последовательностей не образует закономерного ряда.

Пусть

$$'(\alpha \oplus i) = \alpha_i \quad (i = 1,\ 2,\ \ldots,\ n);$$
$$'(\beta \oplus i) = \beta_i \quad (i = 1,\ 2,\ \ldots,\ n);$$
$$'c = n \ominus 1;\ '\varphi = \alpha \oplus n;\ '\psi = \beta \oplus n.$$

Адресная программа

$$\begin{array}{c} Ц\,\{'c\,(-1)\,0 \to ц\} \\ {}^{2}('\varphi \ominus '\text{ц}) \to '\,('\psi \ominus '\text{ц}) \\ Я. \end{array}$$

Программа для машины «Урал»

```
        30 c
        25 0000
R . . . 30 φ
        —30 0000
        02 0000
        30 ψ
        —30 0000
        16 0000
        24 R
        30 ω
        22 0000.
```

Приведем для сравнения программу для машины «Урал», в которую в явном виде входят параметры α, β и n:

$$
\begin{array}{rl}
 & 25\ n \ominus 1 \\
R\ \ldots & -30\ \alpha \oplus n \\
 & 02\ 0000 \\
 & -30\ \beta \oplus n \\
 & 16\ 0000 \\
 & 24\ R \\
 & 30\ \omega \\
 & 22\ 0000.
\end{array}
$$

В качестве упражнения рекомендуем читателю составить для всех приведенных в примерах адресных программ эквивалентные им адресные программы, реализуемые машиной «Урал».

Пример 7. *Умножение матрицы с неполным заполнением A на вектор X* $Y = A \times X$ Запишем программу умножения квадратной матрицы с неполным заполнением, обеспечивающую умножение лишь тех элементов матрицы, которые не равны нулю, без затраты времени на проверку равенства их нулю.

Занумеруем элементы матрицы по строкам номерами 1, 2, ..., n^2, где n — порядок матрицы, и введем адреса лишь для элементов, отличных от нуля (в порядке их следования по строкам)

$$\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha \oplus N.$$

Здесь число N ($N \leqslant n^2$) равно числу элементов исходной матрицы, отличных от нуля.

Для сохранения информации о принадлежности ненулевого элемента матрицы определенному столбцу и строке введем адреса

$$\beta \oplus 1,\ \beta \oplus 2,\ \ldots,\ \beta \oplus N,$$

в которые поместим соответствующие порядковые номера отличных от нуля элементов.

Введем адреса для заданного вектора X

$$\gamma \oplus 1,\ \gamma \oplus 2,\ \ldots,\ \gamma \oplus n$$

и адреса для результирующего вектора Y

$$\tau \oplus 1,\ \tau \oplus 2,\ \ldots,\ \tau \oplus n.$$

Положим, что вначале $'(\tau \oplus i) = 0$ ($i = 1, 2, \ldots, n$). Занумеруем строки матрицы (сверху вниз) и ее столбцы (слева направо) номерами 1, 2, ..., n.

Пусть $'u$ пробегает значения $N \ominus 1$, $N \ominus 2$, ..., 0.

Очевидно, что по адресу $\beta \oplus N \ominus 'u$ может быть определен номер строки и столбца, содержащегося по адресу $\alpha \oplus N \ominus 'u$. Обозначим через i и j функции, по которым определяются номера строки и столбца элемента $'(\alpha \oplus N \ominus 'u)$

$i = i('u)$ — номер строки элемента $'(\alpha \oplus N \ominus 'u)$;

$j = j('u)$ — номер столбца элемента $'(\alpha \oplus N \ominus 'u)$.

Между функциями i, j и функциями «целая часть», «дробная часть» (ц. ч. и д. ч.) существует следующая связь:

$$i('(\beta \oplus N \ominus 'u)) = \begin{cases} \text{ц. ч. } \dfrac{'(\beta \oplus N \ominus 'u)}{n} + 1, & \text{если д. ч. } \dfrac{'(\beta \oplus N \ominus 'u)}{n} \neq 0; \\ \text{ц. ч. } \dfrac{'(\beta \oplus N \ominus 'u)}{n}, & \text{если д. ч. } \dfrac{'(\beta \oplus N \ominus 'u)}{n} = 0; \end{cases}$$

$$j\ ('(\beta \oplus N \ominus 'u)) = \begin{cases} n \times \text{д.ч.} \dfrac{'(\beta \oplus N \ominus 'u)}{n}, & \text{если д. ч. } \dfrac{'(\beta \oplus N - 'u)}{n} \neq 0; \\ n, & \text{если д. ч. } \dfrac{'(\beta \oplus N - 'u)}{n} = 0. \end{cases}$$

Согласно правилу умножения матрицы на вектор, элемент матрицы a_{ij} должен быть умножен на $'(\gamma \oplus j)$ и результат умножения прибавлен к содержимому адреса $\tau \oplus i$.

Таким образом, имеем адресную программу

А неполная Х...

$$\begin{gathered} \boldsymbol{Ц}\{N \ominus 1\,(--1)\,0 \rightarrow u\},\ \mathcal{B} \\ '(\alpha \oplus N \ominus 'u) \times '(\gamma \oplus j\,('(\beta \oplus N \ominus 'u))) + '(\tau \oplus i\,('(\beta \oplus N \ominus \\ \ominus 'u))) \rightarrow \tau \oplus i\,('(\beta \oplus 'u)). \end{gathered}$$

При тех же предположениях относительно исходного отображения адресов (их заполнения) составим программу для машины «Урал».

Предварительно заметим, что номера i (строки) и j (столбца) элемента $'(\alpha \oplus N \ominus 'u)$ могут быть получены повторным вычитанием из $'(\beta \oplus N \ominus 'u)$ числа n до получения первого отрицательного или нулевого результата. Число вычитаний будет равно номеру строки i, а последний положительный (не равный нулю) остаток — номеру столбца j.

При вычитании равных значений на машине «Урал» результат вычитания будет иметь отрицательный знак (этот результат равен —0), а следовательно, если $'(\alpha \oplus N \ominus u)$ — последний элемент строки, то адрес $\beta \oplus N \ominus 'u$ содержит число nd, кратное n, и в результате последовательных вычитаний из него числа n отрицательный результат будет получен после d вычитаний; таким образом, последний неотрицательный остаток будет равен n — номеру столбца.

Для вычисления номера строки введем адрес c (счетчик числа вычитаний). Пусть $'\delta = n$. Получаем адресную программу вычисления функций i и j

$$\begin{gathered} 0 \rightarrow c \\ R\ \ldots\ '(\beta \oplus N \ominus 'u) - '\delta \rightarrow S \\ \boldsymbol{P}\{'S \leqslant 0\}\,R1 \\ 'S \rightarrow \beta \oplus N \ominus 'u \\ 'c \oplus 1 \rightarrow c \\ R \\ R1\ \ldots\ldots\ldots\ldots \end{gathered}$$

В результате работы данной программы имеем

$$\begin{gathered} 'c = i \\ '(\beta \oplus N \ominus 'u) = j. \end{gathered}$$

Отсюда получаем программу для машины «Урал» (в условных адресах)

$l \oplus 0000$ ... $25\ n \ominus 1$
1 ... 20 0000
2 ... $-16\ \tau \oplus n$
3 ... $24\ l \oplus 0001$
4 ... $25\ N \ominus 1$
5 ... 20 0000
6 ... $16\ c$
7 ... $-02\ \beta \oplus N$
$l \oplus 0010$ ... $03\ \delta$
11 ... $21\ l \oplus 0017$
12 ... $-16\ \beta \oplus N$
13 ... 20 0001
14 ... $01\ c$
15 ... $16\ c$
16 ... $22\ l \oplus 0007$
17 ... $-30\ \beta \oplus N$
$l \oplus 0020$ ... 02
21 ... $-06\ \alpha \oplus N$
22 ... $30\ c$
23 ... 01
24 ... $30\ c$
25 ... $16\ \tau \oplus 1$
26 ... $24\ l \oplus 0005$
27 ... 37

ГЛАВА VII

ПРИМЕРЫ И УПРАЖНЕНИЯ

Настоящая глава посвящена разбору более сложных задач и составлению алгоритмов их решения. Значительная часть этих примеров возникла из практики решения задач на ЭВЦМ.

При выборе этих задач преследовалась цель дать возможность читателю приобрести более твердые навыки в составлении алгоритмов: две задачи, имеющие арифметический характер, связаны с обозреванием массивов информации (обработка информации по данной функции веса; табулирование кусочно-постоянной функции); одна задача экономического содержания (расчет плана производства по заданной программе выпуска); одна задача касается оптимальной организации производства (составление графиков загрузки производственного участка); одна — из области линейного программирования; одна задача геометрическая (о склеивании квадрата), связана с проблемой раскладки и одна задача логико-информационного содержания (поиск значения по таблице). В конце главы приведены упражнения для самостоятельной работы.

1. РАСЧЕТ ПЛАНА ПРОИЗВОДСТВА ПО ЗАДАННОЙ ПРОГРАММЕ ВЫПУСКА

При составлении народнохозяйственного плана часто приходится решать следующую задачу. Имеется N видов продукции, закодированных номерами $n = 1, 2, \ldots, N$. Для единицы каждого вида (конечного) продукции i ($1 \leqslant i \leqslant N$) согласно определенной технологии имеются нормы прямых затрат всех видов (исходной) продукции, задаваемые в виде таблиц норм.

Для рассматриваемой замкнутой системы производства указаны задания той части конечной продукции, которая не участвует в производстве в планируемый период (не используется как исходная). Эта продукция может идти на потребление внутри системы для пополнения запасов либо в другие районы. Требуется определить общий план производства по всем видам продукции, обеспечивающий выполнение указанного задания [1].

Обозначим через x_i — план производства по i-му виду продукции; k_{ij} — количество продукции j-го вида, идущей непосредственно на производство единицы продукции i-го вида; a_i — задание на выпуск продукции i-го вида (исключая производственные нужды). Тогда для определения x_i получим систему линейных уравнений

$$x_i = \sum_{j=1}^{N} k_{ij} x_j + a_i \quad (i = 1, 2, \ldots, N).$$

Если обозначить

$$\| k_{ij} \| = K, \quad \begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix} = \bar{a}; \quad \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} = \bar{x},$$

то в матричной записи получим

$$(E - K)\, x = a.$$

Известно, что матрица коэффициентов прямых затрат имеет норму меньше 1. Следовательно, ряд

$$\bar{x} = \bar{a} + K\bar{a} + K^2\bar{a} + \ldots,$$

в котором каждая компонента вектора $\bar{x}$ состоит из суммы всевозможных произведений вида

$$a_i k_{ii_1} k_{i_1 i_2} \ldots k_{i_m l_i},$$

при любом $\bar{a}$ сходится.

Назовем i-м элементом таблицы норм потребления совокупность ее строк, относящихся к единице i-го элемента плана выхода задания (в таблицу включаются лишь

---

[1] Здесь рассматривается задача расчета потребностей без учета их распределенности во времени.

строки, соответствующие не равным нулю коэффициентам матрицы K):

i	j_1	k_{ij_1}
	j_2	k_{ij_2}
. . .	. . .	. . .
	. . j_{l_i}	$k_{ij_{l_i}}$

Здесь в первой строке указывается номер i-продукта. Число строк в этой таблице зависит от i, т. е. от вида продукции; не исключено, что на изготовление j-й продукции расходуется некоторое количество этой же продукции, т. е., что для некоторого j_s

$$i = j_s.$$

Пусть дан итоговый список элементов плана задания, упорядоченный по номерам видов продукции,

$$\begin{array}{ll} 1 & b_1 \\ 2 & b_2 \\ & \vdots \\ N & b_N, \end{array}$$

где b_i — объем i-го вида продукции (если какой-либо из видов продукции i не входит в задание, то соответствующее b_i равно нулю).

Считаем, что

$$'(m \oplus i) = b_i$$

$$'m = N \ \ (N \text{ — число элементов плана}).$$

Пусть элементы таблиц норм потребления упорядочены также по номерам продукции и заданы в виде единой таблицы; тогда вместо задания в первых строках элементарных таблиц норм потребления номеров продукции будем отмечать их специальным признаком; всю совокупность строк элементарных таблиц разместим по адресам в γ-последовательности следующим образом:

$'(\gamma \bigoplus i)_0 = \begin{cases} 1, & \text{если } i\text{-ая строка — первая в элементарной} \\ & \text{таблице норм;} \\ 0, & \text{в противном случае;} \end{cases}$

$'(\gamma \bigoplus i)_1 = n_s$ — соответствующий номер продукции, помещенной в i-й строке общей таблицы норм потребления; $'(\gamma \bigoplus i)_{11}$ — ее объем.

$'\gamma = M$ — общее число строк в таблице норм.

Введем в рассмотрение α и β-последовательности по N адресов в каждой. Последовательность α назовем α-таблицей приращений, или просто α-таблицей; вначале в нее перенесем таблицу плана; последовательность β назовем таблицей накоплений, или β-таблицей; вначале все ее адреса содержат нули, кроме $'\beta = N$.

Для расчета общего плана производства в принятых обозначениях может быть предложен следующий алгоритм. Последовательно обозреваются элементы α-таблицы — неудовлетворенные потребности. Если величина i-й потребности $'(\alpha \bigoplus i)$ превышает некоторую малую величину — допустимую погрешность,— то осуществляется следующий алгоритм удовлетворения потребности.

Потребность (содержимое адреса $\alpha \bigoplus i$) прибавляется в соответствующую строку таблицы накоплений, умножается на соответствующие коэффициенты таблицы норм, и полученные произведения после предварительной засылки нуля на место удовлетворенной потребности (по адресу $(\alpha \bigoplus i)$) прибавляются к соответствующим строкам таблицы приращений. Если $'(\alpha \bigoplus i) < \varepsilon$, то строка в α-таблице пропускается. Таблица приращений обозревается повторно до тех пор, пока все потребности с точностью до ε не будут удовлетворены [1].

Сходимость процесса обеспечивается тем, что в β-таблице накапливаются произведения вида $a_j \cdot k_{jl_1} \ldots k_{jm^{l_i}}$ только в другом порядке, а сходимость абсолютно сходящегося ряда не зависит от перестановки его членов.

В данном алгоритме можно принять следующий порядок обозревания информации: одновременно обозревать γ-последовательность, начиная с адреса $\gamma \bigoplus 1$, и α-таблицу, начиная с адреса α. Если при этом обнаружим, что соответствующая строка таблицы норм является первой строкой элементарной таблицы, то переходим к обработке следующей строки в α-таблице. В противном случае продол-

[1] В приведенном виде алгоритм был сообщен автору Н. З. Шором.

жаем обработку той же строки в α-таблице. При этом элементы α-таблицы обрабатываются, если они превышают некоторое число ε. Для проверки малости всех элементов таблицы приращений используем адрес r, по которому перед каждым повторным циклом обработки засылаем нуль, а при появлении элемента, превышающего ε,— единицу. Нуль по адресу r в конце цикла обработки всей γ-таблицы служит признаком конца работы алгоритма. В принятых обозначениях искомый алгоритм может быть записан в виде:

Расчет потребностей ... M ... $0 \Rightarrow r$, $0 \Rightarrow \pi 1$

$$\boldsymbol{Ц}\,\{1\,(1)'\,\gamma \Rightarrow \pi\}\,R$$
$$'(\gamma \bigoplus '\pi)_1 \Rightarrow r3$$
$$\boldsymbol{P}\,\{'(\gamma \bigoplus '\pi)_0 = 1\} \downarrow M2$$
$$'\pi 1 \bigoplus 1 \Rightarrow \pi 1$$
$$M2 \ldots '(\alpha \bigoplus '\pi 1) + '(\beta \bigoplus '\pi 1) \Rightarrow \beta \bigoplus '\pi 1$$
$$'(\alpha \bigoplus '\pi 1) \Rightarrow r4;\ 0 \Rightarrow \alpha \bigoplus '\pi 1$$
$$'(\gamma \bigoplus '\pi)_{11} \times 'r4 + '(\alpha \bigoplus 'r3) \Rightarrow \alpha \bigoplus 'r3$$
$$\boldsymbol{P}\,\{'(\alpha \bigoplus 'r3 < \varepsilon\}\,M1 \downarrow$$
$$1 \Rightarrow r$$
$$M1 \ \ldots \ R \ \ldots \ \boldsymbol{P}\,\{'r = 0\}\,! \downarrow M$$

Результатом работы алгоритма является содержимое β-последовательности — таблица плана производства всех видов продукции для выполнения указанного задания.

2. ВЫЧИСЛЕНИЕ СУММЫ ЗНАЧЕНИЙ КУСОЧНО-ПОСТОЯННОЙ ФУНКЦИИ

Составить программу вычисления таблицы значений функции

$$s(a, n) = \frac{(b_1 \cdot \sum\limits_{j=n+a}^{n+50} R(j) + b_2 \cdot \sum\limits_{j=n+52}^{n+a+50} R(j)) \cdot \sum\limits_{j=a}^{n+a} K(j)}{b_3 \sum\limits_{j=n+1}^{n+a} R(j) + b_4 K(n) + b_5 R(n)}$$

для всевозможных сочетаний аргументов a и n:

$$a = 1,\ 2,\ \ldots,\ 50;$$
$$n = 1,\ 2,\ \ldots,\ N - 100 \qquad (N > 100)$$

с запоминанием ее в последовательности адресов $s \bigoplus 1$, $s \bigoplus 2$, ...

Здесь b_1, b_2, b_3, b_4, b_5 — заданные константы, $'bs = b_s$; $s = 1, 2, \ldots, 5$; $K(j)$ и $R(j)$ — кусочно-постоянные

функции целочисленного аргумента с совпадающими интервалами постоянства, для которых заданы:

а) $'\varphi = k$ — число интервалов постоянства функций;

б) концы интервалов постоянства $'(m \oplus i) = m_i$ $(i = 1, 2, \ldots, k)$;

в) R_i и K_i $(i = 1, 2, \ldots, k)$ — значения функций $R(j)$ и $K(j)$ на этих интервалах: $'(K \oplus i) = K_i$; $'(R \oplus i) = R_i$.

Концы интервалов выбраны так, что при $i = m_i$

$$K_j = K(i+1);\ R_j = R(i+1).$$

Составим вначале подпрограмму для вычисления значений суммы вида $\sum_{j=t}^{T}$, которую пометим меткой *Сигма*.

В качестве формальных параметров для этой подпрограммы принимаем:

t — нижняя граница параметра j;

T — верхняя граница параметра j;

Q — адрес, по которому в зависимости от суммируемой функции засылается адрес K или R;

d — адрес, по которому подпрограмма выдает результат.

Таким образом, формула вхождения на подпрограмму *Сигма* будет иметь вид:

$$П\,Сигма\{t,\ T,\ Q,\ d\}. \qquad (7.1)$$

Примем следующий порядок работы. Последовательно обозреваем концы интервалов m_i. Если m_i лежит вне интервала $[t, T]$, то переходим к обозреванию следующего конца. Если $m_i \in [t, T]$, то подсчитываем число точек i-го интервала постоянства, попадающих в интервал $[t, T]$, и путем умножения его на соответствующее значение функции (равное $'('Q \oplus i)$) накопляем суммы и так до тех пор, пока не окажется, что $m_i > T$. Согласно формуле вхождения (7.1) подпрограмма *Сигма* может быть записана в виде

```
Сигма ... ∅ ⇒ t; ∅ ⇒ T; ∅ ⇒ Q; ∅ ⇒ d
          t ⇒ r; 0 ⇒ 'd
          Ц{1 (1)'φ ⇒ i} L
          Р{'(m ⊕ 'i) ≤ t} L1
          Р{'(m ⊕ 'i) > T} L
          ²d + ('(m ⊕ 'i) ⊖ 'r) × '('Q ⊕ 'i) ⇒ 'd
          '(m ⊕ 'i) ⇒ r
L1 ... Ь ... ('T − 'r + 1) × '('Q ⊕ 'i) + ²d ⇒ 'd; Я.
```

Заметим, что при $m_{i-1} < {}'t \leqslant m_i$ и ${}'t = {}'T$ эта подпрограмма выдает значение функции, содержащееся по адресу ${}'(Q \oplus i)$.

Необходимая программа теперь может быть записана в виде

$$s \ldots 0 \Rightarrow i$$
$$Ц\{1\,(1)\,50 \Rightarrow a\}$$
$$Ц\{1\,(1)\,{}'N1 \Rightarrow n\}\,M$$
$${}'i \oplus 1 \Rightarrow i$$
$$П\ Сигма\ \{{}'n \oplus {}'a,\ {}'n \oplus 50,\ R,\ d1\}$$
$$П\ Сигма\ \{{}'n \oplus 52,\ {}'n \oplus {}'a \oplus 50,\ R,\ d2\}$$
$$П\ Сигма\ \{{}'a,\ {}'n \oplus {}'a,\ K,\ d3\}$$
$$П\ Сигма\ \{{}'n \oplus 1,\ {}'n \oplus {}'a,\ R,\ d4\}$$
$$П\ Сигма\ \{{}'n,\ {}'n,\ K,\ d5\}$$
$$П\ Сигма\ \{{}'n,\ {}'n,\ R,\ d6\}$$
$$({}'b1 \times {}'d1 + {}'b2 \times {}'d2) \times {}'d3 : ({}'b3 \times {}'d4 + {}'b4 \times {}'b5 + b5 \times {}'d6) \Rightarrow s \oplus {}'i$$
$$M \ldots !$$

3. СХЕМА ОБРАБОТКИ ИНФОРМАЦИИ ПО ЗАДАННОЙ ТАБЛИЦЕ ВЕСОВ [1]

Дана прямоугольная матрица значений функции $u(i, j)$ в точках целочисленной квадратной сетки $i = 0, 1, \ldots, n$; $j = 0, 1, \ldots, m$ и квадратная матрица коэффициентов (таблица весов) c_{kl}; $k, l = -10, -9, \ldots, 10$; $m, n > 21$.

Составить программу вычисления таблицы значений функции

$$s(x, y) = \sum_{j=y-10}^{y+10} \sum_{i=x-10}^{x+10} c_{i-x,\ i-y} u_{ij}$$

для всех $x = 10\,(1)\,{}'N$, $({}'N = {}'n \ominus 10)$; $y = 10\,(1)\,{}'M$, ${}'M = m \ominus 10$ с выдачей результатов на печать.

А. Пусть матрицы $u(i, j)$ и $c(k, l)$ заданы по строкам в виде u- и c-последовательностей:

$${}'(u \oplus (i \ominus 1) \otimes m \oplus j) = u_{ij};$$
$${}'(c \oplus (k \ominus 1) \otimes 21 \oplus l) = c_{kl}.$$

---

[1] Схемы рассматриваемого вида встречаются, например, при обработке геофизических наблюдений.

Искомый алгоритм может быть записан в виде

$$B \ \ldots \ \boldsymbol{Ц}\{10\,(1)'\,M \Rightarrow y\}\,My$$
$$\boldsymbol{Ц}\{10\,(1)'\,N \Rightarrow x\}\,Mx$$
$$0 \Rightarrow s$$
$$\boldsymbol{Ц}\{'x \ominus 10\,(1)'\,x \oplus 10 \Rightarrow i\}\,Mi$$
$$\boldsymbol{Ц}\{'y \ominus 10\,(1)'\,y \oplus 10 \Rightarrow j\}\,Mj$$
$$'s + '(c \oplus ('i \ominus 'x) \otimes 21 \oplus j \ominus 'y) \otimes '(u \oplus ('x \ominus 1)' \otimes m \oplus 'y) \Rightarrow s$$
$$Mj \ \ldots \ Mi \ \ldots \ П_ч's;\ Mx \ \ldots \ My \ \ldots \ Я.$$

Б. Рассмотрим предыдущий пример в предположении, что коэффициенты матрицы c_{kl} симметричны относительно центрального c_{00} и кодируются (по строкам) лишь неповторяющиеся ее элементы, а именно те, для которых

$$(0 \leqslant k \leqslant 10) \wedge (0 \leqslant l \leqslant k),$$

а результаты запоминаются в z-последовательность.

Заметим, что при $k \neq 0$, $r \neq 0$, $k \neq r$ коэффициент c_{kl} при вычислении каждого из значений $v(x, y)$ будет использован 8 раз посредством умножения на следующие значения функции u:

$$u_{x \pm k,\ y \pm l};$$
$$u_{x \pm l,\ y \pm k}.$$

Очевидно, что при $k = 0$ или $l = 0$, или $k = l$ таких значений будет четыре и при $k = l = 0$ — одно.

Для упрощения записи алгоритма разделим соответствующие элементы таблицы треугольной матрицы весов на 4 и на 8. Тогда, полагая, что эта матрица задана по строкам, искомый алгоритм запишется в виде

$\boldsymbol{B}$ ...

$$0 \Rightarrow z$$
$$\boldsymbol{Ц}\{11\,(1)\,'n \ominus 10 \Rightarrow x\}\,Mx$$
$$\boldsymbol{Ц}\{11\,(1)\,'m \ominus 10 \Rightarrow y\}\,My$$
$$0 \Rightarrow s$$
$$\boldsymbol{Ц}\{0\,(1)\,10 \Rightarrow i\}\,Mi$$
$$'z \oplus 1 \Rightarrow z$$
$$u \oplus 'y \Rightarrow r1$$
$$'r1 \oplus ('x \oplus 'i \ominus 1) \otimes 'm \Rightarrow r2$$
$$'r1 \oplus ('x \ominus 'i \ominus 1) \otimes 'm \Rightarrow r3$$
$$'i \otimes ('i \oplus 1) \ :2 \Rightarrow r4$$

$$\textit{Ц}\{0\,(1)\,'i \Rightarrow k\}\,Mk$$
$$'r1 \oplus ('x \oplus 'k \ominus 1) \otimes 'm \Rightarrow r5$$
$$'r1 \oplus ('x \ominus 'k \ominus 1) \otimes 'm \Rightarrow r6$$
$$'('r2 \oplus 'k) + '('r2 \ominus 'k) + '('r3 \ominus 'k) + '('r3 \oplus 'k) +$$
$$+ '('r5 \oplus 'i) + '('r5 \ominus 'i) + '('r6 \ominus 'i) + '('r6 \ominus 'i) \Rightarrow$$
$$\Rightarrow r5;\ 'r4 \oplus 'k \Rightarrow r6;$$
$$'s + '(c \oplus 'r6) \times 'r5 \Rightarrow s \oplus 'z$$
$$Mk\ \ldots\ Mi\ \ldots\ My\ \ldots\ Mx\ \ldots\ \textit{Я}.$$

4. РЕШЕНИЕ ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ[1]

Приведенный метод решения задачи линейного программирования основан на применении одной из R-операций, введенных в работе [13].

I. R-конъюнкция. Конъюнкцией двух чисел a и b называется число $c = (a + b - \sqrt{a^2 + b^2}) : 2$. Обозначение ее $c = a \wedge b$. Приведем следующие свойства R-конъюнкции:

1°. $a \wedge b = b \wedge a$;

2°. $a \wedge b > 0$, тогда и только тогда, когда $a > 0$ и $b > 0$;

3°. $a \wedge b = 0$, тогда и только тогда, когда $a \geqslant 0$, $b = 0$ или $a = 0$, $b > 0$;

4°. Если $a < 0$, то $a \wedge b < 0$ при произвольном b.

Из свойств R-конъюнкции следует, что если (S_1) и (S_2) — области в n-мерном пространстве, определяемые неравенствами $f_1(x_1, \ldots, x_n) \geqslant 0$, $f_2(x_1, \ldots, x_n) \geqslant 0$, то область (S), определяемая неравенством $f_1 \wedge f_2 \geqslant 0$, есть пересечение (общая часть) областей (S_1) и (S_2).

Кроме того, если в каждой точке открытой области (S) n-мерного пространства существуют вторые производные функций $f_1(x_1, \ldots, x_n)$ и $f_2(x_1, \ldots, x_n)$ по любому направлению l и всегда

$$\frac{\partial^2 f_1}{\partial l^2} \leqslant 0;\ \frac{\partial^2 f_2}{\partial l^2} \leqslant 0,\ \text{то}\ \frac{\partial^2}{\partial l^2}[f_1 \wedge f_2] \leqslant 0.$$

2. Отделение множества планов. Задача линейного программирования может быть сформулирована следующим образом.

[1] Предположение о линейности неравенств используется для упрощения выкладок. Этот метод может быть применен и при решении других задач математического (нелинейного) программирования.

Найти решение системы неравенств

$$\begin{aligned}
&L_1 \equiv x_1 \geqslant 0\\
&L_2 \equiv x_2 \geqslant 0\\
&\dots\dots\dots\dots\dots\dots\dots\dots\\
&L_n \equiv x_n \geqslant 0\\
&L_{n+1} \equiv a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n - b_1 \geqslant 0 \qquad (7.2)\\
&L_{n+2} \equiv a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n - b_2 \geqslant 0\\
&\dots\dots\dots\dots\dots\dots\dots\dots\\
&L_{n+m} \equiv a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n - b_n \geqslant 0,
\end{aligned}$$

минимизирующее функцию («функцию цели»)

$$z = c_1x_1 + c_2x_2 + \dots + c_nx_n. \qquad (7.3)$$

Всякое решение системы неравенств (7.2) называется планом. Множество [1] планов представляет собой некоторую выпуклую область в n-мерном пространстве. План, минимизирующий функцию (7.3), называется оптимальным.

Пусть $F(x_1, x_2, \dots, x_n)$ — функция, определяемая следующим образом:

$$F \equiv (((\dots ((L_1 \wedge L_2) \wedge L_3) \dots) \wedge L_{m+n-1}) \wedge L_{m+n}. \qquad (7.4)$$

Из приведенных свойств R-конъюнкции следует, что множество планов определяется неравенством

$$F(x_1, \dots, x_n) \geqslant 0. \qquad (7.5)$$

Уравнение $F = 0$ — уравнение поверхности (S_0), отделяющей выпуклую область планов (σ_0) от остальных точек n-мерного пространства.

Таким образом, задача линейного программирования сводится к нахождению решения неравенства (7.5), минимизирующего функцию (7.3).

3. Семейство гиперповерхностей. Рассмотрим семейство гиперповерхностей, определяемых уравнением

$$F(x_1, \dots, x_n) = c = \text{const}. \qquad (7.6)$$

При $c = 0$ уравнение (7.6) определяет поверхность (S_0), ограничивающую выпуклую область планов (σ_0). На осно-

[1] Множество называется выпуклым, если из принадлежности точек A и B к этому множеству следует принадлежность к нему всех точек отрезка AB.

вании неравенства (7.5) положительным значениям c соответствуют гиперповерхности, лежащие всеми своими точками в области (σ_0), а отрицательным значениям c — гиперповерхности, лежащие вне (σ_0). Можно показать, что каждая из гиперповерхностей $F(x_1, \ldots, x_n) = \text{const}$ есть граница некоторой выпуклой области.

4. Алгоритм вычисления F и grad F. Предположим, что известны верхняя и нижняя оценки для функции цели: $\mu \leqslant z \leqslant M$. Тогда выполняется неравенство

$$L_{m+n+1} = -c_1x_1 - c_2x_2 - \ldots - c_nx_n + M \geqslant 0. \quad (7.7)$$

Присоединим неравенство (7.7) к системе неравенств (7.2). Пусть $F^*(x_1, \ldots, x_n)$ — функция, построенная тем же способом, что и функция $F(x_1, \ldots, x_n)$, но с учетом неравенства (7.7).

Область планов (σ_0^*), определяемая неравенством $F^* \geqslant 0$, или совпадает с областью (σ_0), определяемой неравенством $F \geqslant 0$, или составляет часть области (σ_0).

Пусть точка $M(x_1^0, \ldots, x_n^0)$ лежит вне области планов (σ_0^*). Тогда по крайней мере для одного значения i, $1 \leqslant i \leqslant m + n + 1$

$$L_i < 0. \quad (7.8)$$

Обозначим первое из значений $i = 1, 2, \ldots, m + n + 1$, для которого выполняется условие (7.8) через k.

Для вычисления F^* и grad F^* в точке M, лежащей вне области планов (σ_0^*), используем рекуррентный процесс. Положим

$$\left.\begin{array}{l} F_1 = L_k < 0; \quad A_i = \sqrt{F_{i-1}^2 + L_{i'}^2}; \\ F_i = F_{i-1} \wedge L_{i'} = \dfrac{1}{2}(F_{i-1} + L_{i'} - A_i) \\ (i = 2, 3, \ldots, n + m + 1); \\ F^* = F_{m+n+1}(x_1^0, \ldots, x_n^0). \end{array}\right\} \quad (7.9)$$

Здесь i' пробегает последовательно все значения $i = 1, 2, \ldots, m + n + 1$, исключая значение $i = k$. Поскольку $F_1 = L_k < 0$ вследствие свойства 2° R-конъюнкции, ни одно из F_i $(i = 1, 2, \ldots, m + n + 1)$ не будет обращаться в нуль.

Из формул (7.9) последовательно находим

$$\left.\begin{aligned} &\frac{\partial F}{\partial l} = \frac{\partial L_k}{\partial l}; \\ &\frac{\partial F_i}{\partial l} = \frac{(A_i - F_{i-1})\frac{\partial F_{i-1}}{\partial l} + \frac{\partial L_{i'}}{\partial l}(A_i - L_{i'})}{2A_i} \\ &\qquad (i = 2,\ 3,\ \ldots,\ m+n+1). \end{aligned}\right\} \qquad (7.10)$$

Так как ни одно из $F_i \neq 0$, то знаменатель в (7.10) при всех $i = 2,3,\ \ldots,\ m+n+1$ будет отличным от нуля.

Для построения вектора

$$\bar{N} = \operatorname{grad} F^* = \frac{\partial F}{\partial x_1}\tau_1 + \frac{\partial F}{\partial x_2}\tau_2 + \ldots + \frac{\partial F}{\partial x_n}\tau_n$$

необходимо определить $\frac{\partial F}{\partial x_k}(k = 1,\ 2,\ \ldots,\ n)$. Для этого в формулы (7.10) вместо $\frac{\partial L_i}{\partial l}$ $(i = 1,\ 2,\ \ldots,\ m+n+1)$ подставим $\frac{\partial L_i}{\partial x_j}$, учитывая, что

$$\frac{\partial L_i}{\partial x_j} = \begin{cases} 0 \text{ при } i \neq j,\ i \leqslant n \\ 1 \text{ при } i = j,\ i \leqslant n \end{cases} (j = 1,\ 2,\ \ldots,\ n);$$

$\frac{\partial L_i}{\partial x_j} = a_{ij}$ при $i = n+1,\ n+2,\ \ldots,\ n+m+1$ $(j = 1,\ 2,\ \ldots,\ n)$.

Таким образом, вектор $\bar{N} = \operatorname{grad} F^*$ получим в результате построения последовательности векторов $\left(\frac{\partial F_i}{\partial x_1},\ \ldots,\ \frac{\partial F_i}{\partial x_n}\right.$, $i = 1,\ 2,\ \ldots,\ m+n+1)$ по рекуррентным формулам (7.10), которые можно применять одновременно с формулами (7.9).

При построении алгоритма предполагаем, что задача решена, если оптимальный план найден с некоторой точностью ε, т. е. если найден такой план, для которого значение функции цели равно z, причем $z - z_0 < \varepsilon$ (z_0 — точное значение).

Зададимся произвольной точкой $A_0(x_1^0,\ \ldots\ x_n^0)$ и определим ее принадлежность области планов (σ_0^*). Если $A_0 \in (\sigma_0^*)$, то вычислим значение функции цели $z(A_0)$ и заменим в формуле (7.7) M на $z(A_0) - \varepsilon$. Такая замена приведет к замене области планов (σ_0^*) на область (σ^*), по отношению к которой точка A_0 будет внешней точкой. От точки A_0

перейдем в точку A_1, сделав единичный шаг[1] в направлении grad $F^*(A_0)$,

$$\overline{N}_1 = \frac{\operatorname{grad} F^*(A_0)}{|\operatorname{grad} F^*(A_0)|}.$$

Зная координаты точки A_k и компоненты вектора $\overline{N}_k (k = 1, 2, \ldots)$, применяем следующий алгоритм:

1) определим принадлежность точки A_k области (σ^*). Для этого вычислим $L_i(A_k)$ $(i = 1, 2, \ldots, m + n + 1)$ и определим первое из значений $i = l$, для которого $L_l(A_k) < 0$. Если такое l существует, то переходим к п. 3, в противном случае — к п. 2;

2) находим значение функции цели $z(A_k)$; если $z(A_k) \geqslant \mu$, то вместо числа M в формуле (7.7) принимаем число $z(A_k) - \zeta$ и переходим к п. 3. Если $z(A_k) < \mu$, то процесс прекращаем: задача не имеет решения;

3) если $|\overline{N}_i| \geqslant \varepsilon$, переходим к точке A_{k+1}, прибавляя к координатам точки A_k компоненты вектора $\overline{N}_{k+1}$, определяемые по формулам

$$\overline{N}_{k+1} = \begin{cases} \dfrac{|\overline{N}_k|\operatorname{grad} F^*(A_k)}{|\operatorname{grad} F^*(A_k)|}, & \text{если } \overline{N}_k \cdot \operatorname{grad} F^*(A_k) \geqslant 0; \\ \dfrac{|\overline{N}_k|\operatorname{grad} F^*(A_k)}{2|\operatorname{grad} F^*(A_k)|}, & \text{если } \overline{N}_k \cdot \operatorname{grad} F^*(A_k) < 0, \end{cases}$$

и переходим к п. 1. Если $|\overline{N}_i| < \varepsilon$, то процесс прекращается; координаты точки A_k составляют оптимальный план.

Адресное отображение

Зададим исходную информацию по фиксатору ψ в виде единой β-последовательности (к началу работы алгоритма $'\psi = \beta$) в следующем порядке:

m; n; коэффициенты a_{ij} $(i = 1, 2, \ldots, m + 1; j = 1, 2, \ldots, n)$ в порядке их записи по строкам (причем $a_{i+1,\,j} = c_j$); свободные члены b_j $(j = 1, 2, \ldots, m + 1; b_{m+1} = M)$; μ; x_i^0 $(i = 1, 2, \ldots, n)$ — нулевые приближения (всего $mn + 2n + m + 4$ адресов).

Последние n адресов в алгоритме, в которых размещаются x_i^0, используются для хранения соответствующих величин L_i $(i = 1, 2, \ldots, n)$. Непосредственно за исходной

[1] В учебных целях здесь приведена упрощенная схема выбора шага.

информацией размещаются рабочие массивы в следующем порядке:

$$L_{n+j}\,(j = 1,\ 2,\ \ldots,\ m+1);$$
$$\frac{\partial}{\partial x_i}\ (i = 1,\ 2,\ \ldots,\ n);$$
$$|N|;\ N_i\ (i = 1,\ 2,\ \ldots,\ n).$$

Кроме этого, в алгоритме используется малое положительное число ε (заданная точность) и некоторое малое отрицательное число ζ (для поиска такого l, для которого $L_l<0$). Обозревание отдельных подпоследовательностей в алгоритме осуществляется с помощью дополнительных фиксаторов, которые формируются в начале его работы

Адресный алгоритм

$''\psi \Rightarrow \beta 1;\ '('\psi \oplus 1) \Rightarrow \alpha 1;\ '\beta 1 \oplus '\alpha 1 \oplus 1 \Rightarrow \omega 3$
$'\psi \oplus '\beta 1 \otimes '\alpha 1 \oplus '\alpha 1 \oplus 1 \Rightarrow \omega;\ '\omega \oplus '\beta 1 \oplus 1 \Rightarrow \omega 1$
$'\omega 1 \oplus 1 \Rightarrow \varphi;\ '\varphi 1 \oplus '\alpha 1 \Rightarrow \omega 2;\ '\omega 2 \oplus '\beta 1 \oplus 1 \Rightarrow \delta$
$'\delta \oplus '\alpha 1 \oplus 1 \Rightarrow \rho;\ '\beta 1 \oplus 1 \Rightarrow \beta 2$

H ... $1 \Rightarrow \xi$

$H1$ ... $\boldsymbol{Ц}\{1\,(1)\,'\beta 2 \Rightarrow \pi 1\}\,q$
$'('\omega \oplus '\pi 1) \Rightarrow '\omega 2 \oplus '\pi 1$
$\boldsymbol{Ц}\{1\,(1)\,'\alpha 1 \Rightarrow \pi 2\}$
$'('\omega 2 \oplus '\pi 1) + '('\psi \oplus ('\pi 1 \ominus 1)\,'\alpha 1 \oplus '\pi 2) \times$
$\times\ '('\varphi \oplus '\pi 2) \Rightarrow '\omega 2 \oplus '\pi 1$
q ...
$\boldsymbol{Ц}\{1\,(1)\,'\omega 3 \Rightarrow \pi\}\,q1$
$'\pi \Rightarrow \eta$
$\boldsymbol{Р}\{'('\varphi \oplus '\pi) \leqslant \zeta\}\,M10$
$q1$ ...
$^2\omega 1 - {}^2\delta \Rightarrow s$
$\boldsymbol{Р}\{'s \geqslant {}^2\varphi\}\,'s - '\tau \Rightarrow \omega 1;\ H1 \downarrow !$ [нет решения]

$M10$ ... $'('\varphi \oplus '\eta) \Rightarrow F1$
$\boldsymbol{Р}\{'\eta \leqslant '\alpha 1\} \downarrow M11$
$\boldsymbol{Ц}\{1\,(1)'\alpha 1 \Rightarrow \pi\}$
$0 \Rightarrow '\delta \oplus '\pi$
$1 \Rightarrow '\delta \oplus \eta$
$M12$

$M11 \ldots \; \boldsymbol{Ц}\{1\,(1)'\alpha 1 \Rightarrow \pi\}$
$'('\psi \bigoplus ('\eta \ominus '\alpha 1 \ominus 1) \otimes '\alpha 1 \bigoplus '\pi) \Rightarrow '\delta \bigoplus '\pi$

$M12 \ldots \; \boldsymbol{Ц}\{1\,(1)'\omega 3 \Rightarrow \pi 1\}\, M5$
$\boldsymbol{Р}\{'\pi 1 = '\eta\}$
$'('\varphi \bigoplus '\pi 1) \Rightarrow z;\ \sqrt{'F1^2 + 'z^2} \Rightarrow r1$
$('F1 + 'z - 'r1) : 2 \Rightarrow F2;\ 'r1 - 'F1 \Rightarrow r$
$'r1 - 'z \Rightarrow r3;\ 2 \times 'r1 \Rightarrow r1$
$\boldsymbol{Р}\{'\pi 1 \leqslant '\alpha 1\} \downarrow M4$
$\boldsymbol{Ц}\{1\,(1)'\alpha 1 \Rightarrow \pi 2\}\ M3$
$'r \times ('(\delta \bigoplus '\pi 2) \Rightarrow r2$
$\boldsymbol{Р}\{'\pi 1 = '\pi 2\}\, 'r2 + 'r3 \Rightarrow r2$
$'r2 : 'r1 \Rightarrow '\delta \bigoplus '\pi 2$

$M3 \ldots \; 'F2 \Rightarrow F1$
M

$M4 \ldots \; \boldsymbol{Ц}\{1\,(1)\,'\alpha 1 \Rightarrow \pi 2\}$
$('r \times '('\delta \bigoplus '\pi 2) + '('\psi \bigoplus ('\pi 1 \ominus '\alpha 1 \ominus 1) \otimes$
$\otimes '\alpha 1 \bigoplus '\pi 2) \times 'r3) : 'r1 \Rightarrow '\delta \bigoplus '\pi 2$
$M3$

$M \ldots M5 \ldots \; \boldsymbol{Р}\{'\xi = 1\} \downarrow M14$
$0 \Rightarrow \xi,\ 0 \Rightarrow s$
$\boldsymbol{Ц}\{1\,(1)\,'\alpha 1 \Rightarrow \pi\}$
$'('\delta \bigoplus '\pi)^2 + 's \Rightarrow s$
$\sqrt{'s} \Rightarrow s;\ 1 \Rightarrow '\rho$
$\boldsymbol{Ц}\{1\,(1)\,'\alpha 1 \Rightarrow \pi\}\, q7$
$'('\delta \bigoplus '\pi) : 's \Rightarrow '\rho \bigoplus '\pi$
$'('\varphi \bigoplus '\pi) + '('\rho \bigoplus '\pi) \Rightarrow '\varphi \bigoplus '\pi$
$q7 \ldots H1$

$M14 \ldots \; \boldsymbol{Р}\{^2\rho \leqslant '\tau\}\, M9$
$0 \Rightarrow s;\ 0 \Rightarrow s1$
$\boldsymbol{Ц}\{1\,(1)\,'\alpha 1 \Rightarrow \pi\}\, q10$
$'('\delta \bigoplus '\pi)^2 + 's \Rightarrow s$
$'('\rho \bigoplus '\pi) \times '('\delta \bigoplus '\pi) + 's1 \Rightarrow s1$
$q10 \ldots \sqrt{'s} \Rightarrow s$
$\boldsymbol{Р}\{'s1 > 0\}\, M8$
$2 \times 's \Rightarrow s$

$M8 \ldots 0 \Rightarrow s1$

$\boldsymbol{Ц}\{1\,(1)\,'\alpha 1 \Rightarrow \pi\}\,q11$

$^2\rho \times '('\delta \oplus '\pi) : 's \Rightarrow '\rho \oplus '\pi$

$'('\varphi \oplus '\pi) \oplus '('\rho \oplus '\pi) \Rightarrow '\varphi \oplus '\pi$

$'s1 + '('\rho \oplus '\pi)^2 \Rightarrow s1$

$q11 \ldots \sqrt{'s1} \Rightarrow '\rho$ i $H1$

$M9 \ldots \Pi ч' \geqslant \varphi$

!

5. Построение графиков загрузки оборудования производственного участка [1]

Рассмотрим следующую задачу календарного планирования. Имеется производственный участок (цех) с $'s$ рабочими местами, оснащенными оборудованием определенного типа. На этот участок поступают партии с различным количеством деталей определенных наименований — номеров i ($i = 1, 2, \ldots, I$); не исключено поступление нескольких партий деталей одного и того же наименования.

Каждая деталь i характеризуется своим технологическим маршрутом $Tm\,(i)$ — некоторой упорядоченной последовательностью операций O_{ij}, полностью определяющей очередность и длительность прохождения деталями рабочих мест. Каждая j-я операция технологического маршрута i-й детали O_{ij} определяется своей технологической привязкой l_{ij} к рабочему месту K_{ij} или группе оборудования, которая указывает, на каком рабочем месте может быть выполнена данная операция, величину подготовительно-заключительного времени η_{pj} — часть времени обработки всей партии p на условном единичном рабочем месте, не зависящую от величины партии, и величину штучного времени обработки детали τ_{ij} (норму времени обработки одного изделия i-го вида детали по j-й операции).

Таким образом,

$$Tm\,(i) = \{<j,\ l_{ij},\ \eta_{pj},\ \tau_{ij}>,\ j = 1,\ 2,\ \ldots,\ n_i\}.$$

---

[1] Постановка задачи и общая схема решения предложены автору В. В. Шкурбой. При составлении данного алгоритма была использована схема представления информации для более упрощенной задачи, решенной А. Н. Пшичук.

Назовем графиком $g(p)$ обработки p-й партии i-ых деталей упорядоченную последовательность векторов

$$Q(p,\ j) = < j,\ K_{pj},\ t_{pj},\ T_{pj} > \ (j = 1,\ 2,\ \ldots,\ n_p),$$

где p — номер партии;
n_p — число операций в ее маршруте;
j — номер операции;
K_{pj} — номер рабочего места, на котором выполняется эта j-я операция;
t_{pj} — время начала обработки;
T_{pj} — время ее продолжительности, определяемое равенством

$$T_{pj} = \eta_{pj} + \tau_{pj} \cdot \frac{N_p}{r_{K_{ij}}}.$$

Здесь N_p — величина партии;
i — номер детали в ней;
$r_{K_{ij}}$ — коэффициент производительности рабочего места K_{ij} при соблюдении соотношений

$$\left.\begin{array}{l} t_{pj} + T_{pj} + c_{pj} \leqslant t_{pj+1} + T_{pj+1}; \\ t_{pj} + c_{pj} \leqslant t_{pj+1}, \end{array}\right\} \qquad (7.11)$$

где c_{pj} — некоторая константа, зависящая от времени транспортировки и величины τ_{pj+1}.

Условия (7.11) означают, что j-я операция в партии должна начинаться и заканчиваться несколько ранее $j+1$-й операции, хотя допускается их параллельное выполнение.

План-графиком производственного участка называется множество графиков $g(p)$ обработки партий p ($p = 1,\ 2,\ \ldots,\ n$), если для любых Q_{pj} и Q_{xy} выполняется условие неперекрытия времени использования на одном и том же рабочем месте, т. е. при $K_{pj} = K_{xy}$ неравенство

$$t_{pj} \leqslant t_{xy} \leqslant t_{pj} + T_{pj}$$

невозможно. Перечисленные требования определяют план-график работы участка с так называемой последовательно-параллельной обработкой деталей.

Приведем алгоритм, по которому, задаваясь описанием маршрутов при различном наборе некоторых заданных параметров, можно строить план-графики работы производственных участков по различным критериям.

Назовем пролеживанием p-й партии после j-й операции промежуток времени

$$v_{pj} = t_{pj+1} + T_{pj+1} - t_{pj} - T_{pj} - c_{pj}.$$

Определим допустимое время пролеживания как некоторую задаваемую по каждой партии p константу V_p. Для минимизации производственного цикла эта константа определяется так называемым идеальным графиком обработки партии, т. е. таким графиком $g(p)$, для которого все $v_{pj} = 0$ (для достижения более плотной загрузки можно полагать величину V_p очень большой).

Введем для партий p признаки важности (предпочтения) σ_p. Определим понятие накопленного времени пролеживания партии $Д_{pj}$ к моменту начала выполнения операции $j + 1$

$$Д_{pj} = \sum_{i=1}^{j} v_{pi}$$

и относительного времени пролеживания u_{pj} партии p к моменту начала $j + 1$-й операции

$$u_{pj} = \frac{Д_{pj} n_p}{V_{pj}}.$$

Будем считать, что накопленное время пролеживания «терпимо»[1], если $u_{pj} \leqslant 1$.

Предполагается, что все оборудование (рабочие места) разделено на группы рабочих мест, взаимозаменяемых по ряду операций. Операции, которые могут выполняться на любом рабочем месте в группе, называются привязанными к группе. Допускаются операции, которые могут выполняться только на конкретном рабочем месте (индивидуальная привязка), и операции, выполняемые вне участка (буферные операции).

Рабочие места в группах разделяются на две категории — основные и резервные. Резервные рабочие места используются только в том случае, когда пролеживание по данной операции при попытке ее закрепления за ос-

[1] Несмотря на кажущуюся близость понятий допустимого пролеживания и важности они далеко не эквивалентны. Так, для партии, время обслуживания которой невелико, допустимое время пролеживания может быть значительным наряду с высоким признаком важности.

новными рабочими местами группы превосходит некоторую константу $'cc$, которую для простоты принимаем одинаковой для всех операций. К моменту составления календарного плана на участок могут поступать совсем необрабатывавшиеся или частично обработанные партии.

Для удобства в дальнейшем различные данные будем размещать по таблицам, принимая следующие обозначения. Пусть μ — некоторая таблица. Через μ_{I}, μ_{II} и т. д. будем обозначать соответственно ее столбцы. Строки μ-таблицы будем предполагать заданными как содержимые последовательности адресов $\mu \oplus 1$, $\mu \oplus 2$, ..., римской цифрой на месте индекса в адресе i-й строки $\mu \oplus i$ будем обозначать адрес соответствующего столбца таблицы. Таким образом, $'(\mu \oplus i)$ обозначает содержимое всей i-й строки в μ-таблице, а $'(\mu \oplus i)_{\mathrm{I}}$ — содержимое 1-го столбца в i-й строке и т. д.

Для составления графика загрузки участка о каждой из поступающих партий в виде γ-таблицы по столбцам задаются следующие данные:

γ_{I} — количество необработанных операций по данной партии;

γ_{III} — номер партии p;

γ_{II} — ее объем;

γ_{IV} — номер детали (i);

γ_{V} — дата, ранее которой партия не может быть запущена в обработку (дата запуска) (t_{p1});

γ_{VI} — признак важности σ_p;

γ_{VII} — допустимое пролеживание V_p.

Информация для маршрутов деталей (для данного вида деталей она фиксирована) хранится в виде некоторых α- и β-таблиц. Таблица β называется таблицей заголовков, таблица α — таблицей маршрутов.

Таблица α-маршрутов представляет собой совокупность таблиц индивидуальных маршрутов деталей. В каждой j-й строке индивидуального i-го маршрута указывается:

α_{I} — величина штучного времени обработки i-го вида детали по j-й операции (τ_{ij});

α_{II} — признак привязки данной операции к оборудованию — групповая или индивидуальная привязка;

α_{III} — номер рабочего места в случае индивидуальной привязки или первого рабочего места в группе в случае групповой привязки.

Последние два элемента в таблице α назовем привязкой операции.

В i-ой строке таблицы β содержится следующая информация об i-й детали:

β_I — о месте хранения маршрута i-й детали в таблице маршрутов α — адрес первой строки следующего элементарного маршрута в α-таблице;

β_{II} — общее количество операций в маршруте данной детали (n_i).

Информация о рабочих местах представляется в виде ν-таблицы, содержащей в k-й строке

ν_I — коэффициент производительности r_K рабочего места K;

ν_{II} — номер следующего места в группе, если это место не последнее в группе, и нуль — в противном случае;

ν_{III} — время его освобождения;

ν_{IV} — время переналадки этого места для каждой новой операции (ради простоты это время считается независимым от вида операций и объема партии[1]);

ν_V — признак предпочтения.

Отбор варианта план-графика загрузки оборудования из множества возможных план-графиков осуществляется по следующим правилам.

Операции закрепляются за рабочими местами в порядке их очередности в маршрутах соответствующих видов деталей.

Из претендующих на одно и то же место операций за рабочим местом закрепляется та, для которой

1) t_{pj} — минимально (первый признак предпочтения — загрузка оборудования);

2) при равных t_{pj} меньше σ_p (это уменьшает производственный цикл по партиям, имеющим признак важности);

3) при равных σ_p и $u_{pl} > 1$ u_{pj} — больше (это выравнивает сроки ожидания по одинаково важным деталям);

4) при $u_{pj} \leqslant 1$ с наименьшим $t_{pj} + T_{pj}$ (временем окончания).

Конечная информация представляется в виде следующих трех таблиц:

графика загрузки каждого рабочего места;

графика движения каждой партии;

таблицы сменных заданий на планируемый период.

---

[1] Такое допущение в данном случае не является существенным.

Выберем следующий порядок работы. Введем три рабочие таблицы ρ, δ и ω.

δ — таблица конкурентов, т. е. данных о первых стоящих на очереди операциях партий. Число строк в этой таблице n равно числу всех подлежащих раскладке партий; i-я строка соответствует i-й партии и хранит о ней следующую информацию в соответствующих столбцах δ-таблицы:

δ_{I} — норма продолжительности данной операции ($\tau_{pj} \times \times N_p$);

δ_{II} — продолжительность предыдущей операции этой же детали (T_{pj-1});

δ_{III} — накопленное пролеживание рассматриваемой детали ($Д_{pj}$);

δ_{IV}, δ_{V} — технологическая привязка (как и в α-таблице α_{II} и α_{III});

δ_{VI} — время начала предыдущей операции (t_{pj-1});

ρ — таблица претендентов (условных загрузок). Число строк в ней соответствует количеству рабочих мест на участке; l-я строка соответствует l-му из них.

В процессе работы алгоритма в соответствующих столбцах этой таблицы накапливается следующая информация о претендующей операции:

ρ_{I} — продолжительность операции (T_{pj});

ρ_{II} — начало операции (t_{pj});

ρ_{III} — номер соответствующей строки в таблице δ или γ (номер p закрепляемой партии);

ρ_{IV} — относительное пролеживание;

ω — таблица реальных привязок, в которой по столбцам размещается следующая информация о каждой привязанной операции всех маршрутов;

ω_{I} — реальная продолжительность операции (T_{pj});

ω_{II} — время ее начала (t_{pj});

ω_{III} — номер обрабатываемой детали (i);

ω_{IV} — номер партии (p);

ω_{V} — номер данной операции в маршруте (j);

ω_{VI} — номер рабочего места, к которому эта операция привязывается K_{pj}.

После полной раскладки всего плана по информации в ω-таблице строятся графики загрузки рабочих мест и движения партий и таблица сменных заданий на планируемый период.

Для заполнения δ-таблицы информацией об очередных операциях введем подпрограмму *Выбор К* — выбор конкурента — с формулой вхождения

$$П\ Выбор\ К\ \{l\},$$

где l — означает номер строки δ- или γ-таблицы (номер партии).

$$Выбор\ К\ \ldots\ \varnothing \Rightarrow \lambda$$
$$'('\gamma \oplus '\lambda) \Rightarrow \xi;\ '\delta \oplus '\lambda \Rightarrow \eta$$
$$M2\ \ldots\ \boldsymbol{P}\{'\xi_{I} > '('\xi_{IV} \oplus '\beta)_{II}\}\, 0 \Rightarrow '\eta;\ \textit{Я}$$
$$'('('\xi_{IV} \oplus '\beta)_{I} \oplus '\xi_{I}) \Rightarrow z$$
$$'\xi_{II} \times 'z_{I} \Rightarrow '\eta_{I};\ '\xi_{I} + 1 \Rightarrow ('\gamma \oplus '\lambda)_{I}$$
$$\boldsymbol{P}\{'z_{III} \neq 0\}\, 'z_{II,\ III} \Rightarrow '\eta_{IV,\ V};\ \textit{Я}$$
$$^{2}\eta_{I} + {}^{2}\eta_{II} + '\xi_{V} \Rightarrow (\gamma \oplus '\lambda)_{V};\ 0 \Rightarrow '\eta_{III};\ M2$$

Здесь адреса ξ, η, z введены для сокращения записи; как и в соответствующих таблицах принято:

$$'\xi_{I} = '(\gamma \oplus '\lambda)_{I} = j;\ '\xi_{II} = n_{j}$$

и т. д.

В строке с меткой $M2$ проверяется, закреплены ли все операции в партии с номером $'\lambda$ за рабочими местами, и засылается нуль в соответствующую строку таблицы δ, если партия полностью закреплена.

В следующей строке, если в партии имеются незакрепленные операции, то по номеру детали $'(\gamma \oplus '\lambda)_{IV} = '\xi_{IV}$ и очередной операции $'(\gamma \oplus '\lambda)_{I} = \xi_{I}$ из таблицы заголовков маршрутов определяется адрес очередной маршрутной строки данной детали. (Эта строка затем засылается в строку z).

В следующей строке вычисляется норма продолжительности данной операции и увеличивается на единицу номер очередной операции в таблице γ.

Далее, проверяется, не является ли данная операция буферной. Для буферных операций ($'z_{III} = 0$) время начала предыдущей операции увеличивается на сумму продолжительности выполнения предыдущей и данной операций и засылается нуль в столбец данной строки δ_{II}, т. е. величина продолжительности предыдущей операции принимается равной нулю.

Затем совершается переход на метку $M2$ и т. д. Информация об операции с привязкой на участке засылается в таблицу δ и совершается выход из подпрограммы.

Таким образом, подпрограмма заканчивает работу либо после выбора операции с привязкой на участке, либо после засылки нуля в строку δ-таблицы, если все операции по данной партии уже закреплены за рабочими местами.

Для заполнения строк таблицы ω введем подпрограмму *Засылка* ω с формулой вхождения

$$П\ Засылка\ \omega\{\lambda a\},$$

где λa — номер строки таблицы ρ (номер рабочего места с минимальным временем начала очередной операции на нем);

μ — номер очередной строки ω-таблицы.

Засылка ω ... $\varnothing \Rightarrow \lambda a$

$$'\rho \oplus '\lambda a \Rightarrow \xi;\ {}^{2}\xi \Rightarrow \xi 3;\ '\omega \oplus '\mu \Rightarrow \eta$$
$$'см — \text{д. ч. } ('\xi 3_{II} : 'см) \Rightarrow раб$$
$$\boldsymbol{P}\{'раб > '\varepsilon\}\ Зб$$
$$\boldsymbol{P}\{'\xi 3_{I} < 2 \times 'раб\}\ 'раб \Rightarrow \xi_{I} \downarrow '\xi 3_{II} + 'раб \Rightarrow \xi_{II}$$
$$Ззб\ \ldots\ '\gamma \oplus '\xi 3_{III} \Rightarrow \xi 1;\ '\delta \oplus '\xi 3_{III} \Rightarrow \xi 2$$
$$'\xi \Rightarrow '\eta;\ '\xi 1_{III,\ IV} \Rightarrow '\eta_{III,\ IV};\ '\xi 1_{I} \Rightarrow '\eta_{V};\ '\lambda a \Rightarrow '\eta_{IV}$$
$$'\xi 3_{II} - '\xi 1_{V} - 'c \Rightarrow h$$
$$\boldsymbol{P}\{'\xi 3_{I} < '\xi 2_{II}\}\ 'h + '\xi 3_{I} - '\xi 2_{II} \Rightarrow h$$
$$'\xi 2_{III} + 'h \Rightarrow '\xi 2_{III};\ '\xi 3_{II} \Rightarrow '\xi 1_{V}$$
$$'\xi 3_{I} + '\xi 3_{II} + ('\nu \oplus '\lambda a)_{IV} \Rightarrow ('\nu + '\lambda a)_{III}$$
$$'\xi 3_{I} \Rightarrow '\xi 2_{II}$$
$$\boldsymbol{П}\ Выбор\ K\{'\xi 3_{III}\}$$
$$0 \Rightarrow '\rho \oplus '\lambda a;\ Я$$
$$Зб\ \ldots\ \text{д. ч. } \{('\xi 3_{I} + '\xi 3_{II}) : 'см\} \Rightarrow раб$$
$$\boldsymbol{P}\{'раб < '\varepsilon\}\ '\xi 3_{I} - 'раб \Rightarrow \xi_{I}$$
$$Ззб$$

По этой подпрограмме заполняются строки ω-таблицы с предварительным округлением времени начала и конца операции с тем, чтобы концы смен не разделяли операции на части, одна из которых (величина *'раб*) имеет относительно малую длительность ($<'\varepsilon$).

Если фиксируемая операция начинается за $'\varepsilon$ единиц времени до конца смены и продолжительность ее ($'\xi_I$) не превосходит удвоенного промежутка времени до окончания смены ($'раб$), то этот промежуток времени принимается в качестве продолжительности данной операции. Таким образом, предполагается, что операция будет выполнена в несколько сжатый срок путем повышения производительности рабочего или использования промежутка времени между сменами (для окончания операции рабочий задержится на работе на этот малый промежуток времени).

Если начинающаяся за $'\varepsilon$ единиц времени до конца смены операция длительная, т. е. продолжительность ее выполнения $'\xi_I$ превосходит величину $2 \times 'раб$, то время ее начала сдвигается на начало следующей смены.

Для операций, время начала которых далеко от времени конца смены ($'раб > '\varepsilon$), проверяется условие, не является ли момент их окончания ($'раб$, вычисляемое в строке с меткой *3б*) незначительно превышающим время окончания смены; при выполнении этого условия величина длительности операции сокращается и за момент окончания ее принимается время конца смены, т. е., как и ранее, длительность операции уменьшается на $'раб$.

Последующие строки, начиная от строки с меткой *3бб*, содержат запись следующих действий:

а) заполнение ω-таблицы;

б) вычисление пролеживания операции ($'h$) на данном рабочем месте; если длительность данной операции превосходит длительность предыдущей, то пролеживание определяется как разность времени начала данной и предыдущей операций, увеличенная на константу $'c$, равную времени транспортировки деталей от одного рабочего места к другому; в противном случае к вычисляемой величине $'h$ прибавляется разность между продолжительностями выполнения данной и предыдущей операций;

в) вычисление накопленного пролеживания и засылка его в соответствующее место δ-таблицы;

г) засылка времени начала данной операции и ее продолжительности, а также времени освобождения рабочего места в соответствующие места γ-, ν- и δ-таблиц;

д) обращение к подпрограмме *Выбор К* (по соответствующему номеру строки $'\xi_{III}$) для заполнения строки δ-таблицы;

е) стирание информации о привязанной операции в соответствующей строке таблицы ρ;

ж) выход из подпрограммы.

Выделим теперь следующие этапы работы алгоритма.

НЗаполнение — подготовка заполнения таблиц δ и ω;

Выбор π — выбор претендентов (заполнение ρ-таблицы);

Закрепление π — закрепление претендентов (заполнение заблицы ω);

Проверка 1 — проверка условия: δ-таблица — непуста;

Проверка 2 — проверка условия: ρ-таблица — непуста;

Построение Г — построение графиков загрузки каждого рабочего места, графиков движения каждой партии и таблицы сменных заданий на планируемый период.

Кратко опишем эти этапы.

Этапу *НЗаполнение* соответствует в приведенном далее алгоритме строка

$$НЗаполнение \ldots 0 \Rightarrow q;\ Выб\ 2\ldots\ 0 \Rightarrow \mu.$$

Здесь q — переключатель, обеспечивающий при первом просмотре — таблицы δ засылку в нее необходимой информации с помощью подпрограммы *Выбор К*; μ — номер строки ω-таблицы.

Алгоритм *Выбор* π представляет собой циклическую программу, на i-м цикле которой обозревается i-я строка δ-таблицы. При первом обзоре δ-таблицы, так как $q = 0$, с помощью формул

$$\begin{array}{l} P\{'q \neq 0\}\ A \\ П\ \ Выбор\ \ К\ \ \{'l\} \\ A\ \ldots \end{array}$$

предварительно каждая ее строка заполняется соответствующей информацией. Вторая из этих формул — обращение на подпрограмму *Выбор К* (буферные операции в таблицу δ подпрограммой *Выбор К* не вносятся).

Обозревание каждой строки δ-таблицы сводится к следующему. Если данная строка содержит нуль, обозревание ее заканчивается (выход по метке $L2$). В противном случае вырабатывается признак — таблица δ непуста ($1 \Rightarrow \tau$; для сокращения записи соответствующая строка таблицы δ засылается по адресу z), выясняется тип привязки и в зависимости от него по адресу $\pi\pi$ засылается нуль или единица. Признаком индивидуальной привязки служит равенство нулю $'z_{1V} = '('\delta \oplus 'p)_{1V}$.

Для операций с групповой привязкой предварительно осуществляется выборка (строки алгоритма от метки *Группа* по метку *L*2) следующей информации:

а) о наличии резервных рабочих мест (засылка нуля или единицы по адресу *рез*);

б) о номере основного рабочего места, освобождающегося ранее остальных основных рабочих мест (по адресу *No*), и времени его освобождения (*'Фо*);

в) о номере резервного рабочего места, освобождающегося ранее других резервных мест (по адресу *Np*), и времени его освобождения (*'Фр*);

и далее номер выбранного основного рабочего места засылается по адресу z_V.

Цикличность работы этого участка алгоритма обеспечивается засылкой $'('_\vee \oplus 'z_V)_{II} \Rightarrow \zeta$ (строка с меткой *Гр*2), в результате которой номер следующего в группе рабочего места принимается за номер данного рабочего места) и последующей за ней предикатной формулой, проверяющей, не является ли данное рабочее место в группе последним (*'гр* — некоторая подходящая константа).

Начиная от строки с меткой *Ка*, по номеру рабочего места $'\zeta$ определяется продолжительность выполнения данной операции на этом рабочем месте (частное от деления нормы продолжительности выполнения операции $'z_I$ на коэффициент производительности рабочего места $'('_\vee \oplus 'z_V)_I)$ и $'d$ — время возможного начала операции (сумма времени окончания предыдущей операции $'('_\gamma \oplus 'I)_V$ и $'c$ — времени доставки деталей к рабочему месту); если продолжительность данной операции меньше продолжительности предыдущей операции той же детали, то время возможного начала увеличивается на величину разности между продолжительностью предыдущей и настоящей операций (данная операция не может быть закончена ранее предыдущей).

Далее по адресу h засылается разность между временем освобождения данного рабочего места $'('_\vee \oplus 'z_V)_{III}$ и временем возможного начала данной операции $'d$. Если эта разность положительна, за возможное время начала выполнения данной операции принимается время освобождения рабочего места. В этом случае деталь пролеживает в ожидании рабочего места. Если указанная разность отрицательна, по адресу h засылается нуль.

Далее по адресу *и* вычисляется накопленное пролеживание данной детали с учетом пролеживания, появляющегося при закреплении операции за данным рабочим местом.

Затем проверяется наличие претендента на данное рабочее место. Если рабочее место свободно, то операция с индивидуальной привязкой привязывается к этому рабочему месту. Для операции с групповой привязкой проверяется, не превосходит ли ее пролеживание на данном рабочем месте некоторую константу (для простоты эта константа *'сс* выбрана общей для всех операций). Если $'h < 'cc$, то к рабочему месту привязывается данная операция; если $'h \geqslant 'cc$, то проверяется выполнение соотношения $'Фр < 'Фо$ (освобождается ли выбранное резервное место ранее выбранного основного) и в зависимости от результата проверки операция привязывается к данному рабочему месту или к резервному. В последнем случае номер выбранного резервного рабочего места засылается по адресу ζ и совершается переход на строку с меткой *Кb*, в которой по адресу $\pi\pi$ засылается нуль. Тем самым привязка резервного рабочего места осуществляется так же, как и при индивидуальной привязке (повторный выход на выбор резервного рабочего места исключается).

Если соответствующая строка таблицы условных привязок заполнена (в ней зафиксирован претендент), то предварительно проверяется, не закреплена ли уже на данном рабочем месте рассматриваемая операция (строка с меткой *М*9). При выполнении этого условия обозревание строки δ-таблицы заканчивается (выход по метке *L*2). В противном случае конкурент сравнивается с претендентом на это рабочее место, в результате которого в таблице либо остается прежний претендент, либо он вытесняется конкурентом. Конкурент с претендентом сравнивается согласно описанным правилам построения графика загрузки (от строки с меткой *Привязка* по строку с меткой *Концы* включительно). В результате полного обзора таблицы δ получается некоторое заполнение (полное или частичное) таблицы ρ.

Алгоритм *Закрепление* π сводится к обозреванию таблицы претендентов ρ и выбору из нее претендента с минимальным временем начала операции (строки от метки *Закрепление* π по метку *L*61). При этом строки,

содержащие нули, опускаются. Если таких претендентов имеется несколько, то выбирается некоторый из них. С помощью подпрограммы *Засылка* ω по номеру выбранного претендента ($'rn$) заполняется строка ω-таблицы (а в соответствующей строке ρ-таблицы информация об этом претенденте стирается; $'гр$ 1 — подходящая константа).

Проверка 1 сводится к предикатной формуле

$$P\{'\tau \neq 0\}\ Выб\ 1,$$

означающей переход к повторному обзору δ-таблицы, если она не пуста, или на алгоритм *Проверка* 2 в противном случае (по адресу τ засылается нуль при каждом новом обращении к алгоритму *Выбор* π).

Проверка 2 так же сводится к одной предикатной формуле

$$P\{'\tau a \neq 0\}\ Закрепление\ \pi \downarrow \boldsymbol{П}\ Построение\ Г,$$

по которой, если таблица ρ не пуста, совершается переход на алгоритм *Закрепление* II и на алгоритм *Построение Г* в противном случае; последнее означает, что закрепление всех операций по всем партиям закончено.

Алгоритм *Построение Г* — построения графиков загрузки состоит из трех частей: построение графиков загрузки каждого рабочего места, графиков движения каждой из обрабатываемых партий и таблицы сменных заданий на планируемый период. Работа этих алгоритмов сводится к выбору соответствующей информации из ω-таблицы; ввиду простоты предоставим их запись осуществить читателю.

$$H\ Заполнение\ \ldots\ 0 \to q$$
$$Выб\ 2\ \ldots\ 0 \Rightarrow \mu$$
$$Выбор\ \pi\ \ldots\ 0 \Rightarrow \tau$$
$$\boldsymbol{Ц}\ \{1\,(1)'\,n \Rightarrow l\}\ Закрепление\ \pi$$
$$\boldsymbol{Р}\{'q \neq 0\}\,A$$
$$\boldsymbol{П}\ Выбор\ К\ \{'l\}$$
$$A\ \ldots\ '('\delta \oplus 'l) \Rightarrow z;\ '('\alpha \oplus 'l) \Rightarrow z1$$
$$'('\gamma \oplus 'l) \Rightarrow z2;\ 'z_{\text{V}} \Rightarrow \zeta$$
$$\boldsymbol{Р}\{'z = 0\}\,L2$$
$$1 \Rightarrow \tau$$
$$P\{'z_{1\text{V}} \neq 0\}\,1 \Rightarrow \pi\pi;\ Группа$$

$$Kb \ \ldots \ 0 \Rightarrow \pi\pi$$

$$Ka \ \ldots \ '(\rho \bigoplus '\zeta) \Rightarrow x; \ '(\nu \bigoplus '\zeta) \Rightarrow y$$

$$'z_{\text{I}} : 'y_{\text{I}} \Rightarrow b; \ 'z2_{\text{V}} + 'c \Rightarrow d$$

$$\boldsymbol{P}\{'b < 'z_{\text{II}}\}\,'d + 'z_{\text{II}} - 'b \Rightarrow d$$

$$'y_{\text{III}} - 'd \Rightarrow h$$

$$\boldsymbol{P}\{'h < 0\}\,0 \Rightarrow h \downarrow 'h + 'd \Rightarrow d$$

$$('('\beta \bigoplus 'z2_{\text{IV}})_{\text{II}} \times ('h + 'z_{\text{III}})) : 'z2_{\text{I}} : 'z2_{\text{VII}} \Rightarrow u$$

$$\boldsymbol{P}\{'x \neq 0\}\,M9$$

$$\boldsymbol{P}\{'\pi\pi = 0\}\ \textit{Привязка}$$

$$\boldsymbol{P}\{'h < 'cc\}\ \textit{Привязка}$$

$$\boldsymbol{P}\{'\ \textit{рез} = 0\}\ \textit{Привязка}$$

$$\boldsymbol{P}\{'\Phi p < '\Phi o\}\,Np \Rightarrow \zeta; \ Kb$$

$$\textit{Привязка} \ \ldots \ 'b_{\text{I}} \wedge 'd_{\text{II}} \wedge 'l_{\text{III}} \wedge 'u_{\text{IV}} \Rightarrow '\rho \bigoplus '\zeta; \ L2$$

$$M9 \ \ldots \ \boldsymbol{P}\{'l = 'x_{\text{III}}\}\,L2$$

$$'d - 'x_{\text{II}} \Rightarrow t$$

$$\boldsymbol{P}\{'t > 0\}\,L2$$

$$P\{'t < 0\}\ \textit{Привязка}$$

$$P\{'z2_{\text{VI}} > '('\gamma \bigoplus 'x_{\text{III}})_{\text{VI}}\}\ \textit{Привязка}$$

$$\boldsymbol{P}\{'z2_{\text{VI}} < '('\gamma \bigoplus 'x_{\text{III}})_{\text{VI}}\}\,L2$$

$$\boldsymbol{P}\{'u \leqslant 1 \wedge 'x_{\text{IV}} \leqslant 1\}\ \textit{Концы}$$

$$\boldsymbol{P}\{'u < 'x_{\text{IV}}\}\,L2$$

$$\boldsymbol{P}\{'u > 'x_{\text{IV}}\}\ \textit{Привязка}$$

$$\textit{Концы} \ \ldots \ \boldsymbol{P}\{'x_{\text{I}} - 'b > 0\}\ \textit{Привязка} \downarrow L2$$

$$\textit{Группа} \ \ldots \ 'rp \Rightarrow \Phi o, \ \Rightarrow \Phi p; \ 0 \Rightarrow \textit{рез}$$

$$\boldsymbol{Ц}\{1\,(1)\,'\Gamma \Rightarrow \pi\}\,L7$$

$$'('\nu \bigoplus '\zeta) \Rightarrow y$$

$$\boldsymbol{P}\{'y_{\text{V}} = 0\}\,\Gamma p\ 1$$

$$\boldsymbol{P}\{'y_{\text{III}} \geqslant '\Phi o\} \downarrow 'y_{\text{III}} \Rightarrow \Phi o; \ '\zeta \Rightarrow No$$

$$\Gamma p2 \ \ldots \ 'y_{\text{II}} \Rightarrow \zeta$$

$$\boldsymbol{P}\{'\zeta \neq 0\}\,L71 \downarrow L7$$

$$\Gamma p\ 1 \ \ldots \ 1 \Rightarrow \textit{рез}$$

$$\boldsymbol{P}\{'y_{\text{III}} \geqslant '\Phi p\} \downarrow 'y_{\text{III}} \Rightarrow \Phi p; \ '\zeta \Rightarrow Np$$

$$\Gamma p2$$

$$L71 \ \ldots$$

$$L7 \ \ldots \ 'No \Rightarrow \zeta; \ Ka$$

$$L2 \ \ldots$$

Закрепление π ... $'rp1 \Rightarrow rp;\ 1 \Rightarrow q$

$\boldsymbol{Ц}\{1\ (1)\ 's \Rightarrow l\}\ L61$

$'('\rho \bigoplus 'l) \Rightarrow y1$

$\boldsymbol{P}\{'y1 = 0\}\ L6 \downarrow 1 \Rightarrow \tau a$

$\boldsymbol{P}\{'y1_{11} < 'rp\}\ 'y1_{11} \Rightarrow rp;\ 'l \Rightarrow \lambda a$

$L6$ ...

$L61$ ... $'\mu \bigoplus 1 \Rightarrow \mu$

$\boldsymbol{П}$ *Засылка* $\omega\ \{'\lambda a\}$

Проверка 1 ... $\boldsymbol{P}\{'\tau \neq 0\}$ *Выб* 1

$\boldsymbol{P}\{'\mu \leqslant 'max\}$ *Проверка* 2

$\boldsymbol{П}$ *Засылка* $\omega\{'rn\}$

$0 \Rightarrow \mu$

Проверка 2 ...

$\boldsymbol{P}\{'\tau a \neq 0\}$ *Закрепление* $\pi \downarrow$ $\boldsymbol{П}$ *Построение* Γ; 1

Выб 1 ... $\boldsymbol{P}\{'\mu \leqslant 'max\}$ *Выбор* π

$\boldsymbol{П}$ *Засылка* $\omega\{'rn\}$

Выб 2

6. ЗАДАЧА О СКЛЕИВАНИИ КВАДРАТА[1]

Квадрат стороны a разрезается несколькими бесконечными прямыми на N составных многоугольников. Последние произвольным образом разбрасываются на плоскости xOy. При этом допускаются переворачивания (зеркальное отображение) многоугольников, а также их взаимное наложение (рис. 3).

Занумеруем составляющие многоугольники номерами 1, 2, ..., N. Назовем первый многоугольник основным, остальные — подклеиваемыми. Будем предполагать, что в качестве исходной информации задаются координаты (x_{ij}, y_{ij}) многоугольников в порядке обхода сторон по часовой стрелке (рис. 4): i — номер многоугольника, j — номер его вершины; $1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant n_i$ (n_i — число вершин в i-м многоугольнике); число N и вектор $\{n_1, n_2, \ldots, n_N\}$ предполагаются также заданными.

---

[1] В разработке алгоритма и машинных программ для этой задачи автору оказали помощь студенты Бердянского пединститута А. Глушко, В. Шкляров, и др., а также Л. Г. Усенко.

Составим адресную программу склеивания исходного квадрата, понимая под склеиванием сборку квадрата из его частей в любом месте плоскости xOy с указанием новых координат вершин всех составляющих многоуголь-

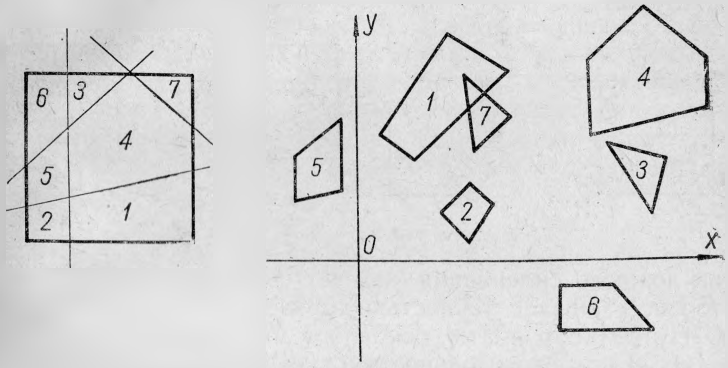


Рис. 3.

ников. Исключим вначале возможность переворачивания многоугольника.

В основу построения алгоритма положим принцип последовательного склеивания многоугольников равными сторонами. Для любого составляющего многоугольника всегда найдется среди остальных по крайней мере один, который можно к нему подклеить, например по рис. 3 к многоугольнику *7* можно подклеить многоугольник *4*; к многоугольнику *5* — многоугольники *2*, *4*, *6*. После склеивания приходим к исходной задаче с числом составляющих многоугольников на единицу меньшим. После каждого склеивания, если оно производится правильно, всегда возможно дальнейшее склеивание вплоть до исчерпания всех составляющих многоугольников. Если же осуществлено ложное склеивание, то либо на некотором шаге исчезает возможность дальнейшего склеивания (рис. 5), либо после пол-

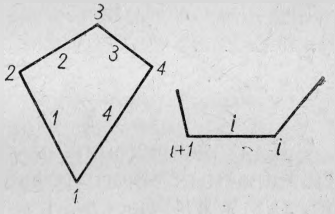


Рис. 4.

ного склеивания получается фигура, отличная от квадрата (рис. 6).

Из-за невысокой точности задания исходной информации (ошибки измерения координат вершин и т. п.) проверку равенства сторон необходимо вести с точностью до некоторого $'\varepsilon$. Это увеличивает число допустимых вариан-

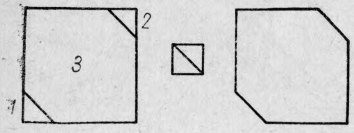


Рис. 5.

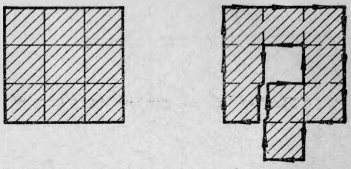


Рис. 6.

тов ложного склеивания за счет возможных склеиваний сторон, равных с точностью до $'\varepsilon$.

Пусть координаты основного многоугольника заданы в $'x1$-$'y1$-последовательностях так, что

$'('x1 \oplus j) = x_{1j}$ — абсцисса j-й вершины основного многоугольника;

$'('y1 \oplus j) = y_{1j}$ — ордината j-й вершины основного многоугольника.

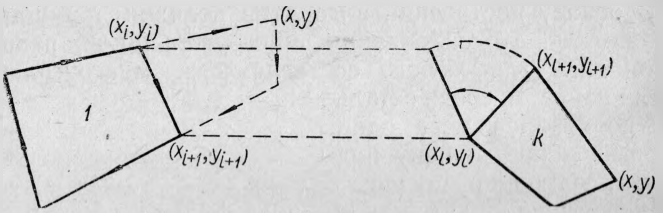


Рис. 7.

Для размещения величин n_i, равных числам сторон склеиваемых многоугольников, выделим $'n$-последовательность: $'('n \oplus i) = n_i$, $1 \leqslant i \leqslant N$. Координаты склеиваемых многоугольников закодируем в $'x$ и $'y$-последовательности:

$x_{ij} = '('x \oplus m_i \oplus j)$ — абсцисса j-й вершины i-го многоугольника;

$y_{ij} = '('y \oplus m_i \oplus j)$ — ордината j-й вершины i-го многоугольника. Здесь $m_i = \sum_{l=2}^{i-1} nl$.

Подклеивание будем осуществлять равными сторонами путем надлежащего перемещения в координатной плоскости подклеиваемых многоугольников к основному (рис. 7). Многоугольник, получаемый в результате склейки, будем снова принимать за основной и соответственно этому изменим нумерацию его вершин. Для этого будем вносить в $'x1$- и $'y1$-последовательности преобразованные координаты присоединяемых вершин так, чтобы сохранялся порядок обхода вершин вновь получаемого многоугольника по часовой стрелке, и соответственно изменению числа вершин в нем изменять $'(n \oplus 1)$. Для этого объемы адресов в $'x1$- и $'y1$-последовательностях считаем равными

$$M = \sum_{i=1}^{N} n_i - 2(N-1),$$

так как при подклеивании k-го многоугольника число вершин в основном увеличивается на $n_k - 2$, а число всех склеиваний равно $N - 1$.

Назовем набором η следующую тройку целых чисел $\{i, k, l\}$:

i — номер стороны основного многоугольника $(1 \leqslant i \leqslant n_1)$;

k — номер одного из подклеиваемых многоугольников $(2 \leqslant k \leqslant N)$;

l — номер одной из сторон k-го многоугольника $(1 \leqslant l \leqslant n_k)$.

Упорядоченное множество наборов η, по которым осуществляются склеивания, назовем μ-множеством.

Каждое μ-множество с числом элементов s, меньшим $N - 1$, определяет упорядоченное множество неподклеенных многоугольников Q с номерами $r_1, r_2, \ldots, r_s$ ($r_i < r$, если $i < j$; $r_s \leqslant N$) и некоторое упорядоченное множество оставшихся наборов $\lambda(\mu)$ с первым набором $\{1, r_1, 1\}$, последним набором $\{N, r_s, n_{r_s}\}$ и операцией следования C, задаваемой следующим описанием:

если $l < n_{r_k}$, то следующим набором за $\{i, r_k, l\}$ считается набор $\{i, r_k, l+1\}$;

если $l = n_{r_k}$ и $k \neq s$, то таким набором считается набор $\{i, r_{k+1}, 1\}$;

если $l = n_{r_s}$, то следующим за набором $\{i, r_k, l\}$ считается набор $\{i+1, r_1, 1\}$.

Назовем остатком $\lambda(\mu, \eta)$ множества $\lambda(\mu)$ упорядоченное множество с операцией следования C, определенной

в множестве $\lambda(\mu)$, с первым элементом $C\eta$, следующим за набором η, и последним, совпадающим с последним элементом множества $\lambda(\mu)$. Начальное расположение соответствует пустому множеству μ и множеству $\lambda(\mu)$, содержащему множество всех возможных наборов. Набор из множества $\lambda(\mu)$, по которому возможно склеивание, назовем допустимым.

Для хранения элементов μ-множества выделим три последовательности по $N-1$ адресу в каждой с началами $'pk \bigoplus 1$, $'pl \bigoplus 1$, $'pi \bigoplus 1$, обозревание которых будем осуществлять с помощью адреса сдвига s (вначале $'s = 0$). Таким образом, при склеивании согласно некоторому допустимому набору $\{i, k, l\}$ содержимое s будет увеличиваться на единицу и по адресам $'pi \bigoplus s$, $'pk \bigoplus s$, $'pl \bigoplus s$ будут засылаться соответственно числа i, k, l.

При обнаружении ложного склеивания (дальнейшая склейка невозможна или полученная фигура — не квадрат) расклеивание осуществляется в следующем порядке: набор η (последний элемент μ-множества), по которому осуществлялось последнее склеивание, исключается из μ-множества, а в качестве множества оставшихся наборов рассматривается остаток $\lambda(\mu, \eta)$.

Это соответствует устранению последней склейки с соответствующим изменением содержимого адреса $'n \bigoplus 1$ и $'x1$-, $'y1$-последовательностей, уменьшению $'s$ на единицу и включению последнего из подклеенных многоугольников в множество Q.

Если исчерпание всех возможностей не приведет к построению квадрата, то это будет свидетельствовать либо о допущении ошибок в кодировании исходной информации, либо о том, что с заданной точностью $'\varepsilon$ квадрат не может быть склеен.

Выделим следующие этапы работы алгоритма.

$T1$ — построение $\lambda(\mu)$-множества после склеивания по некоторому допустимому набору. Начальное расположение многоугольников соответствует данному этапу. При этом $\lambda(\mu)$-множество совпадает с множеством всех возможных наборов.

$T2$ — фиксация допустимого набора η.

$T3$ — склеивание по допустимому набору η.

$T4$ — выдача результата и остановка $\mathcal{Я}$.

$T5$ — расклеивание (по последнему элементу μ-множества).

$T6$ — построение $\lambda(\mu, \eta)$ множества после расклеивания (по последнему элементу μ-множества).

$P1$ — поиск в λ допустимого набора η.

$P2$ — проверка условия: исчерпаны (подклеены) все многоугольники.

$P3$ — проверка условия: полученная фигура — квадрат.

$P4$ — проверка допустимости расклеивания, т. е. возможности построения квадрата и остановка ! в случае отсутствия решения.

$P5$ — проверка условия: $\lambda(\mu, \eta)$-множество исчерпано.

Граф-схема алгоритма приведена на рис. 8.

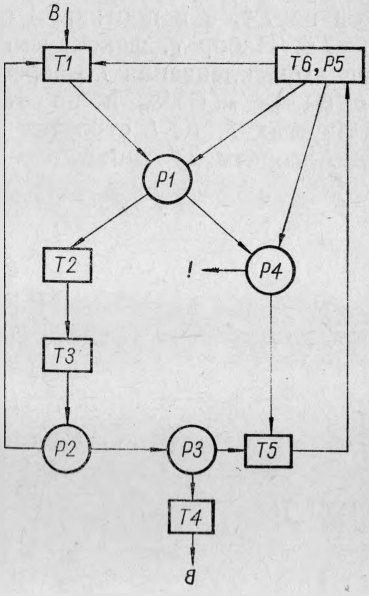


Рис. 8.

Перейдем к описанию перечисленных этапов.

$T1$. Построение множеств $\lambda(\mu)$ и $\lambda(\mu, \eta)$ отличается лишь заданием первого элемента и дальнейшее осуществление алгоритма не зависит от того, в каком из этих множеств осуществляется поиск допустимого набора η. Поэтому поиск η можно осуществить с помощью циклической программы, для которой начальные значения параметров циклов определяются соответственно алгоритмами $T1$ и $T6$.

Для хранения компонент первого набора множеств λ выделим адреса I, K, L. Тогда при построении множества $\lambda(\mu)$ в эти адреса будем засылать числа 1, 2, 1. Алгоритм $T1$ имеет вид

$$T1 \ldots 1 \Rightarrow I$$
$$2 \Rightarrow K$$
$$1 \Rightarrow L$$
$$P1 \ldots$$

$P5$. Проверка условия — множество $\lambda(\mu, \eta)$ непустое — осуществляется программой

$$\boldsymbol{P}\{'L = '('n + 'K)\} \downarrow T6$$
$$\boldsymbol{P}\{'K = 'N\} \downarrow T6$$
$$\boldsymbol{P}\{'I = '('n \oplus 1)\}\, \boldsymbol{P}4 \downarrow T6,$$

согласно которой, если $\lambda(\mu, \eta)$ пустое, совершается переход на $P4$, а в противном случае — на алгоритм $T6$.

$T6$. Набор η, фиксируемый адресом сдвига s, в алгоритме расклеивания $T5$ пересылается по адресам $'pi \oplus 's$, $'pk \oplus 's$, $'pl \oplus 's$, и по этим адресам алгоритмом $T6$ в адресах I, K, L строится начальный набор для $\lambda(\mu, \eta)$.

Алгоритм $T6$ запишется в виде

$$T6 \ \ldots \ P\{'L = '('n \oplus 'K)\}\, 35$$
$$'L \oplus 1 \Rightarrow L$$
$$P1$$
$$35 \ \ldots \ P\{'K = 'N\}\, 36$$
$$'K \oplus 1 \Rightarrow K$$
$$1 \Rightarrow L$$
$$P1$$
$$36 \ \ldots \ P\{'I = '('n \oplus 1)\}\, P4$$
$$'I \oplus 1 \Rightarrow I$$
$$2 \Rightarrow K$$
$$1 \Rightarrow L$$
$$P1$$

Так как условия в предикатных формулах алгоритмов $P5$ и $T6$ совпадают, то эти алгоритмы можно объединить, что и сделано в приведенном ниже алгоритме; кроме того, помечены вторая и третья строки алгоритма $T1$ и сделаны соответствующие сокращения в алгоритме $T6$.

$T2$. Фиксация допустимого набора осуществляется засылками.

$$T2 \ \ldots \ 's \oplus 1 \Rightarrow s;\ 'k \Rightarrow 'pk \oplus 's;\ 'l \Rightarrow 'pl \oplus 's;\ i \Rightarrow 'pi + 's$$
$$T3 \ \ldots$$

$T3$. Алгоритм склеивания осуществляет вставку в $'x1$-$'y1$-последовательности пересчитанных координат вершин k-го многоугольника, исключая координаты вершин стороны, по которой осуществляется подклеивание.

Принятый метод фиксации вершин приводит к тому, что на одной стороне склеенного многоугольника может оказаться несколько вершин (рис. 9), координаты которых содержатся в $'x1$- и $'y1$-последовательностях.

Предварительно вся информация в $'x1$-, $'y1$-последовательностях, начиная с координат $i+1$-й вершины, сдвигается на $n_k - 2$ (т. е. на $'('n \oplus 'k) - 2$) адресов вправо и соответственно изменяется $'('n \oplus 1)$.

Формулы для пересчета координат имеют вид

$$X = (x_{kt} - x_{kl+1}) \frac{x_{kl} - x_{kl+1}}{d} + (y_{kt} - y_{kl+1}) \frac{y_{kl} - y_{kl+1}}{d};$$

$$Y = (y_{kt} - y_{kl+1}) \frac{x_{kl} - x_{kl+1}}{d} - (x_{kl} - x_{kl+1}) \frac{y_{kl} - y_{kl+1}}{d};$$

$$x'_{1i+t} = x_1 + X \frac{x_{1i+1} - x_{1i}}{d} - Y \frac{y_{1i+1} - y_{1i}}{d};$$

$$y'_{1i+t} = y_1 + Y \frac{x_{1i+1} - x_{1i}}{d} + X \frac{y_{1i+1} - y_1}{d},$$

где d — длина склеиваемой стороны, (x_{kt}, y_{kt}) и (x_{1i+t}, y_{1i+t}) — соответственно старые и новые координаты перемещаемой вершины k-го многоугольника (рис. 7).

В результате работы алгоритма $T3$ $'x1$-, $'y1$-последовательности будут содержать информацию о склеенном многоугольнике; при этом порядок записи координат вершин будет соответствовать их обходу по контуру многоугольника по часовой стрелке.

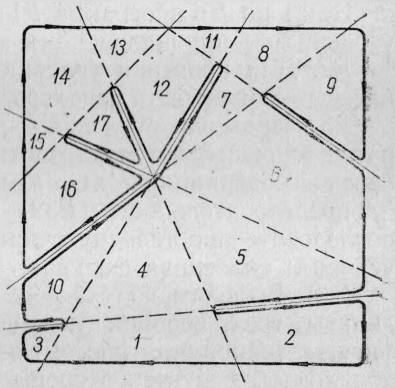


Рис. 9.

$T4$. Этот алгоритм согласно принятому способу кодирования состоит в выдаче на печать (на визуальное выводное устройство) содержимого $'x1$-, $'y1$-последовательностей. Производя разрез многоугольника по ломаной в порядке записи ее вершин, получаем полную картину склеивания квадрата.

$T5$. Алгоритм расклеивания осуществляет по последнему набору η μ-множества засылку набора η по адресам

I, K, L, уменьшение на единицу содержимого адреса сдвига s, сжатие $'x1$-, $'y1$-последовательностей.

$P1$. Поиск допустимого набора η представляет собой циклический перебор элементов множества λ с проверкой свойства: набор η — допустимый. При этом, если η — допустимый набор, то осуществляется выход на $T2$, в противном случае проверяется условие конца поиска. Если поиск не закончен, т. е. η не последний элемент в λ, выбирается следующий элемент в λ и т. д.; если в множестве λ нет допустимого набора, происходит выход на $P4$.

Для определения допустимости набора $\eta\{i, k, l\}$ вычисляются квадраты длин i-й стороны основного и l-й стороны k-го многоугольников и сравниваются между собой с точностью до некоторого $'\varepsilon$. Для этого используется подпрограмма *Квадрат длины* с тремя входами, на первых двух входах указываются адреса абсциссы и ординаты начальной вершины стороны, а на третьем — адрес, по которому подпрограмма выдает результат. Эта подпрограмма приведена в конце алгоритма склеивания квадрата (от строки с меткой *Квадрат длины*).

Цикл по f в программе $P1$ осуществляет выбор многоугольников, принадлежащих множеству Q.

$P2$. Для проверки условия — все многоугольники подклеены—может быть использовано неравенство $'s<'N-1$.

$P3$. Проверка условия — построенная фигура — квадрат — осуществляется алгоритмом из трех частей: $P31$ — перенос содержимого $'x1$-, $'y1$-последовательностей в $'x2$-, $'y2$-последовательности; $P32$ — выбрасывание из $'x2$-, $'y2$-последовательностей внутренних точек и ($P33$) — проверка условия: склеенная фигура — квадрат.

$P32$. Выделим из $'x2$-, $'y2$-последовательностей координаты всех вершин, лежащих на контуре склеенной фигуры (например, рис. 6 и 9, где стрелками указан порядок обхода всех сторон многоугольников, пунктиром отмечены стороны, по которым производилось склеивание), путем выбрасывания координат внутренних точек.

Рассмотрим замкнутую ломаную, координаты вершин которой заданы в $'x2$-, $'y2$-последовательностях.

При правильном склеивании квадрата, если внутри него имеются вершины, все внутренние звенья ломаной и только эти звенья будут сдвоенными. Этим звеньям соответствуют стороны многоугольников, по которым

произошло соприкосновение многоугольников при склеивании каких-либо других сторон. В случае ложного склеивания некоторые из внутренних звеньев могут оказаться несдвоенными (рис. 6).

Таким образом, для выделения граничных точек можно поступить следующим образом. Из трех последовательно взятых вершин, удовлетворяющих условию: две крайние вершины совпадают,— оставим одну из крайних (рис. 9). Выбрасывание с одновременным сжатием информации продолжим до его полного исчерпания. Тогда, в случае правильного склеивания, останутся невыброшенными только вершины на контуре фигуры.

Разделим процесс выбрасывания на два этапа: 1) выбрасывание в разомкнутой цепи вершин согласно кодировке их координат в $'x1$-, $'y1$-последовательностях; 2) выбрасывание на стыке разрыва цепи (рис. 7). Такое выбрасывание возможно, если первыми закодированы координаты внутренней вершины.

Р33. Далее по каждой соседней паре координат вершин, выписанных в $'x2$-, $'y2$-последовательностях, вычисляется угловой коэффициент соответствующей прямой и проверяется условие параллельности или перпендикулярности соседних пар отрезков, начиная с первой пары. Для параллельных пар второй отрезок опускается и с первым сравнивается следующий и т. д. Для ортогональных пар координаты точки пересечения выписываются в $'x3$-, $'y3$-последовательности, а вторая пара принимается за начальную и т. д. В качестве нулевого отрезка берется последний отрезок ломаной.

При обнаружении двух отрезков, не удовлетворяющих (с точностью до $'\varepsilon$) условию параллельности или перпендикулярности, совершается переход на алгоритм $T5$ (построенная фигура — не квадрат). Если такой пары не обнаружено, то проверяется количество вершин с ортогональным пересечением (во избежание построения вида, приведенного на рис. 5). Если таких точек более четырех, осуществляется переход на $T5$ (построенная фигура — не квадрат). Попутно проверяется равенство стороне квадрата $'a$, которая предполагается заданной, расстояний между двумя соседними парами сторон. При выполнении этого условия производится переход на $T4$ — выдачу результатов и остановку, в противном случае — переход на $T5$.

Если в результате перебора все склеивания оказались ложными или допустимый набор вообще отсутствует (это может случиться, например, в результате ошибок, допущенных при кодировании информации), то задача не имеет решения). При реализации алгоритма это соответствует ситуации: необходимо расклеивание, когда склеивания не было.

*P*4. Проверка этого условия осуществляется по содержимому адреса *s*, которое при отсутствии склеиваний равно 0:

$$P4 \ \ldots \ P\{'s = 0\}! \downarrow T5.$$

Дополнительные замечания. Допустим возможность переворачивания (зеркального отображения) составных многоугольников при их разбрасывании на координатной плоскости.

В этом случае приведенный алгоритм необходимо дополнить внешним циклом, в котором при обнаружении невозможности склеивания (после полного перебора всех конфигураций) будет осуществляться переворачивание некоторых из подклеиваемых многоугольников.

Для выполнения полного перебора допустимых переворачиваний поступаем следующим образом. Пусть $N-1$ — число подклеиваемых многоугольников. Тогда различных вариантов размещения подклеиваемых многоугольников будет 2^{N-1}. Рассмотрим отрезок $N-1$-разрядных двоичных чисел от 00 ... 000 до 11 ... 111. Поставим в соответствие каждому i-у разряду этих чисел i-й подклеиваемый многоугольник.

Пусть $a_1a_2 \ \ldots \ a_{N-1}$ — одно из чисел данного множества. Выбирая для подклеивания k-й многоугольник, будем предварительно его отображать, если $a_k = 1$, и сохранять в прежнем виде, если $a_k = 0$. Зеркальное отображение будем осуществлять относительно оси OX; этому соответствует изменение знаков всех ординат многоугольника.

В выбранном множестве двоичных чисел определена операция следования, состоящая в прибавлении единицы; с помощью этой операции следования можно записать формулу циклирования для перебора этих чисел и соответствующего зеркального отображения многоугольников.

Если после полного перебора квадрат не склеен (выход!), то с заданным $'\varepsilon$ задача не может быть решена.

Нетрудно дополнить алгоритм выбором нового $'\varepsilon$ (и так до некоторого максимального значения) и переходом к построению квадрата с точностью до $'\varepsilon$. Идея данного алгоритма может быть использована в задачах о раскрое (раскладка).

Предлагаем читателю дополнить приведенный алгоритм склеивания возможностью зеркального отображения многоугольников и изменения точности.

Склеивание квадрата ... $0 \Rightarrow s;\ \Rightarrow t$

$T1$ ... $1 \Rightarrow I$
2 ... $2 \Rightarrow K$
3 ... $1 \Rightarrow L$

$P1$ ... $Ц\{'I\,(1)'\,('n \oplus 1) \Rightarrow i\}\,40$
$П$ *Квадрат длины* $\{'x1 \oplus 'i,\ 'y1 \oplus 'i;\ d1\}$
$Ц\{'K\,(1)'N \Rightarrow k\}\,17$
$Ц\{1\,(1)'S \Rightarrow f\}\,16$
$P\{'k = '\,('pk \oplus 'f)\}\,37$
$0 \Rightarrow t$
$Ц\{2\,(1)\,('k \ominus 1) \Rightarrow l\}$
$'t \oplus '\,('n \oplus 'l) \Rightarrow t$
$'t \Rightarrow z$
$16...$
$Ц\{'L\,(1)'\,('n \oplus 'k) \Rightarrow l\}\,37$
$'Z \oplus 'l \Rightarrow t$
$П$ *Квадрат длины* $\{'x \oplus 't;\ 'y \oplus 't;\ d\}$
$P\{|\,'d - 'd1\,| > '\varepsilon\} \downarrow T2$
37 ... $1 \Rightarrow L$
17 ... $2 \Rightarrow K$
40 ... $P4$

$T2$ ... $'s \oplus 1 \Rightarrow s;\quad 'k \Rightarrow 'pk \oplus 's;\quad 'l \Rightarrow 'pl \oplus s;$
$'i \Rightarrow 'pi \oplus 's$

$T3$ ... $Ц\{'('n \oplus 1)\,(-1)'i \oplus 1 \Rightarrow t\}\,41$
$'('x1 \oplus 't) \Rightarrow 'x1 \oplus t' \oplus '('n \oplus 'k) \ominus 2$
$'('y1 \oplus 't) \Rightarrow 'y1 \oplus 't \oplus '\,('n \oplus 'k) \ominus 2$
$41...$
$'('n \oplus 1) \oplus '('n \oplus 'k) \ominus 2 \Rightarrow 'n \oplus 1$
$1 \Rightarrow f;\ 'x \oplus 'z \Rightarrow d1;\ 'y \oplus 'z \Rightarrow d2;$

$'x1 \oplus 'i \Rightarrow d3;\ 'y1 \oplus 'i \Rightarrow d4$

$0 \Rightarrow \pi$

$\sqrt{'d} \Rightarrow d;\ (('(d3 \oplus 1) - {}^2d3) : 'd \Rightarrow a1$

$('('d4 \oplus 1) - {}^2d4) : 'd \Rightarrow a2$

$('('d1 \oplus 'l \oplus 1) - '('d1 \oplus 'l)) : 'd \Rightarrow a3$

$('('d2 \oplus 'l \oplus 1) - '('d2 \oplus 'l)) : 'd \Rightarrow a4$

$\boldsymbol{P}\{'l \oplus 2 > '(n \oplus 'k)\}\ 10$

$^{r}l \oplus 2 \Rightarrow \Delta;\ '(n \oplus 'k) \Rightarrow \Delta 1$

12

10 ... $'l \ominus ' ('n \oplus 'k) \oplus 2 \Rightarrow \Delta$

11 ... $'l \ominus 1 \Rightarrow \Delta 1,\ 1 \Rightarrow \pi$

12 ... $\boldsymbol{Ц}\{'\Delta(1)'\ \Delta 1 \Rightarrow t\}\ 42$

$'('d1 \oplus 'l \oplus 1) - '('d1 \oplus 't) \Rightarrow d$

$'('d2 \oplus 'l \oplus 1) - '('d2 \oplus 't) \Rightarrow z$

$'d \times 'a3 + 'z \times 'a4 \Rightarrow X$

$'z \times 'a3 - 'd \times 'a4 \Rightarrow Y$

$^2d3 + 'X \times 'a1 - 'Y \times 'a2 \Rightarrow 'd3 \oplus 'f$

$^2d4 + 'Y \times 'a1 + 'X \times 'a2 \Rightarrow 'd4 \oplus 'f$

$'f \oplus 1 \Rightarrow f$

42...

$\boldsymbol{P}\{'\pi \neq 0\}\ P2$

$\boldsymbol{P}\{'l = 1\}\ P2$

$1 \Rightarrow \Delta;\ 11$

$P2$ ... $\boldsymbol{P}\{'s < 'N \ominus 1\}\ T1$

$'('n \oplus 1) \oplus 1 \Rightarrow z$

$P3$ ... $P31$ ... $\boldsymbol{Ц}\{1(1)'\ z \Rightarrow t\}$

$'('x1 \oplus 't) \Rightarrow 'x2 \oplus 't;\ '('y1 \oplus 't) \Rightarrow 'y2 \oplus 't$

$P32$ ... $\boldsymbol{Ц}\{1(1)'\ z \ominus 2 \Rightarrow t\}\ 44$

14 ... $\boldsymbol{P}\{|'('x2 \oplus 't) - '('x2 \oplus 't \oplus 2)| < '\varepsilon \wedge$

$\wedge |'(y2 \oplus 't) - '('y2 \oplus 't \oplus 2)| < '\varepsilon\} \downarrow 15$

$\boldsymbol{Ц}\{'t \oplus 3(1)'\ z \Rightarrow f\}$

$'('x2 \oplus 'f) \Rightarrow 'x2 \oplus 'f \ominus 2;\ '('y2 \oplus 'f) \Rightarrow$

$\Rightarrow y2 \oplus 'f \ominus 2$

$'z \ominus 2 \Rightarrow z;\ 't \ominus 1 \Rightarrow t;\ 14$

15 ... 44 ...

$16a$ ... $\boldsymbol{P}\{|'('x2 \oplus 2) - '('x2 \oplus 'z \ominus 1)| < 'ε \wedge$
$\wedge |'('y2 \oplus 2) - '('y2 \oplus 'z \ominus 1)| < 'ε\} \downarrow P33$
$'z \ominus 2 \Rightarrow z$
$\boldsymbol{Ц}\{1\,(1)'z \Rightarrow f\}$
$'('x2 \oplus 'f \oplus 1) \Rightarrow 'x2 \oplus 'f;\ '('y2 \oplus 'f \oplus 1) \Rightarrow$
$\Rightarrow 'y2 \oplus 'f$
$16a$

$P33$ ... $0 \Rightarrow t;\ (' ('y2 \oplus 2) - '('y2 \oplus 1)) : (' ('x2 \oplus 2) -$
$- '('x2 \oplus 1) \Rightarrow d3$
$\boldsymbol{Ц}\{2\,(1)'\,z \Rightarrow f\}\ T4$
$(' ('y2 \oplus 'f \oplus 1) - '('y2 \oplus 'f)) : (' ('x2 \oplus$
$\oplus 'f \oplus 1) - ' ('x2 \oplus 'f)) \Rightarrow d4$
$\boldsymbol{P}\{|' d3 - 'd4| < 'ε\}\ 20$
$\boldsymbol{P}\{|' d3 \times 'd4 - 1| < 'ε\} \downarrow T5$
$'d4 \Rightarrow d3;\ 't \oplus 1 \Rightarrow t$
$\boldsymbol{P}\{'t \leqslant 3\} \downarrow 21$
$'('x2 \oplus 'f) \Rightarrow 'x3 \oplus 't;\ '('y2 \oplus 'f) \Rightarrow 'y3 \oplus 't$
$\boldsymbol{P}\{'t = 1\}\ 20$
$\boldsymbol{П}$ *Квадрат длины* $\{'x3 \oplus 't \ominus 1,\ 'y3 \oplus$
$\oplus 't \ominus 1;\ d\}$
$\boldsymbol{P}\{|' d - 'a| < 'ε\}\ 20 \downarrow T5$

21 ... $\boldsymbol{P}\{'t > 4\}\ T5$

20 ... $T4$ ... $Пч'\ \exists x1$
$Пч'\ \exists y1;Я$

$P4$ ... $\boldsymbol{P}\{'s = 0\}!$

$T5$ ... $'('pk \oplus 's) \Rightarrow K;\ '('pi \oplus 's) \Rightarrow I;\ '('pl \oplus 's) \Rightarrow$
$\Rightarrow L$
$'s \ominus 1 \Rightarrow s;\ '('n \oplus 1) \ominus '('n \oplus 'K) \oplus 2 \Rightarrow$
$\Rightarrow 'n \oplus 1$
$\boldsymbol{Ц}\{'I \oplus 1\,(1)'\,('n \oplus 1) \Rightarrow t\}\ 49$
$'('x1 \oplus 't \oplus ' ('n \oplus 'K) \ominus 2) \Rightarrow 'x1 \oplus t;$
$'('y1 \oplus 't \oplus '('n \oplus 'K) \ominus 2) \Rightarrow 'y1 \oplus 't$
49 ...

$T6$ ... $\boldsymbol{P}\{'L = '('n \oplus 'K)\}\ 24$
$'L \oplus 1 \Rightarrow L;\ P1$

$$
\begin{aligned}
24 \ \ldots \ & \boldsymbol{P}\{'K = 'N\}\,25 \\
& 'K \oplus 1 \Rightarrow K;\ 3 \\
25 \ \ldots \ & \boldsymbol{P}\{'I = '('n \oplus 1)\}\,P4 \\
& 'I \oplus 1 \Rightarrow I;\ 2 \\
& \textit{Квадрат длины} \ \ldots \ \varnothing \Rightarrow \xi;\ \varnothing \Rightarrow \eta;\ \varnothing = d \\
& ('('\xi \oplus 1) - ''\xi)^2 + ('('\eta \oplus 1) - ''\eta)^2 \Rightarrow d;\ \text{Я}.
\end{aligned}
$$

7. ПОИСК ЗНАЧЕНИЯ ПО ТАБЛИЦЕ

а) Пусть для значений аргумента

$$x_0 + k \cdot h \ \ (k = 0,\ 1,\ 2,\ \ldots,\ N)$$

заданы значения функции y_k так, что

$$'(\alpha \oplus k) = y_k;\ \ 'x0 = x_0;\ \ 'h1 = h.$$

Требуется по заданному значению x

$$'x1 = x$$

найти ближайшее слева значение x_k и соответствующее значение y_k поместить по адресу s.

Алгоритм вычисления искомого адреса можно записать в виде

$$'(\alpha \oplus \text{ц. ч.}\,(('x - 'x0) : 'h1)) \Rightarrow s.$$

Здесь через ц. ч. обозначена целая часть соответствующей дроби.

При отсутствии в наборе операций машины операции ц. ч. для вычисления целой части дроби $'a : 'b$ можно использовать следующий алгоритм ($'a \geqslant 0,\ 'b > 0$):

$$
\begin{aligned}
& 0 \Rightarrow r \\
& M \ \ldots \ 'a - 'b \Rightarrow a \\
& P\{'a < 0\}\,\text{Я} \downarrow 'r \oplus 1 \Rightarrow r;\ M.
\end{aligned}
$$

б) Пусть в качестве шага $'h1$ выбрано некоторое двоичное число 2^{-m}.

Тогда вычисление по заданному $'x1 = x$ ближайшего слева значения x_k сводится к операции

$$'x1 \wedge '\delta,$$

где $\wedge$ — знак конъюнкции, а по адресу δ содержится константа, имеющая нули в разрядах мантиссы кода числа,

начиная от m — 1-го разряда, и в остальных разрядах — единицы.

Таким образом, получаем алгоритм

$$'(\alpha \ominus 2^m \cdot (x_0\, \theta\, ('x1\ \Lambda\ '\delta))) \Rightarrow s.$$

в) Пусть некоторому множеству слов

$$x_1,\ \ldots,\ x_N \tag{7.12}$$

(словарю множества аргументов) поставлены в соответствие слова

$$y_1,\ \ldots,\ y_N \tag{7.13}$$

(словарь значений функции) так, что

$$\left.\begin{array}{l} '(x \oplus i) = x_i \\ '(y \oplus i) = y_i \end{array}\right\} \quad (i = 1,\ 2,\ \ldots,\ N).$$

Требуется по заданному адресу a слова x_i $(1 \leqslant i \leqslant N)$ поместить соответствующее слово y_i по адресу s. Никаких предположений об упорядоченности множества (7.12) не делается, в связи с чем такую задачу естественно назвать задачей ассоциативного поиска.

Применяя определенную в гл. I § 5 минус-штрих-операцию, искомый алгоритм можем записать в виде

$$'(^{-'}('a\,\{x \oplus 1\,(1)\,x \oplus N\}) \oplus y \ominus x) \Rightarrow s$$

или при $'x = N$

$$'(^{-'}(a \supseteq x) \oplus y \ominus x) \Rightarrow s.$$

Для записи последнего алгоритма без употребления минус-штрих-операции потребуется осуществить поиск слова x в словаре, что можно сделать по программе

$$\begin{array}{l} \textit{Поиск}\ \ldots\ \boldsymbol{Ц}\,\{1\,(1)\,\boldsymbol{Р}\,\{'(x \oplus 'i) \neq 'a\} \Rightarrow i\}\,M \\ \qquad\qquad M\ \ldots\,'(y \oplus 'i) \Rightarrow s;\ \boldsymbol{Я}. \end{array}$$

Имеем пример формулы циклирования с пустой областью действия.

г) Пусть, как и в предыдущем случае, даны множества (7.12) и (7.13), $'x = N$, однако допускается, что среди элементов (7.12) имеются одинаковые. Требуется, используя минус-штрих-операцию, по данному адресу слова x_i заслать в s-последовательность множество всех тех y_i, для

которых $x_i = {}'a$, и по адресу s поместить их число. Искомый алгоритм можно записать в виде

$$\alpha \Rightarrow \varphi,\ 0 \Rightarrow s$$
$$M\ \ldots\ {}^{-'}({}'a\{{}'\varphi \bigoplus 1\,(1)\,\alpha \bigoplus {}'x\}) \Rightarrow \varphi$$
$$\boldsymbol{P}\{{}'\varphi = \varnothing\}!$$
$${}'s \bigoplus 1 \Rightarrow s,\ {}'(y \bigominus x \bigoplus {}'\varphi) \Rightarrow s \bigoplus {}'s$$
$$M.$$

УПРАЖНЕНИЯ

1. Составить алгоритм вычисления значения функции

$$f(x) = \begin{cases} \frac{1}{2}x^2, & \text{если } |x| \geqslant 2; \\ |x|, & \text{если } |x| < 2. \end{cases} \qquad (7.14)$$

2. Составить алгоритм вычисления значений функции (7.14) для всех $x = x_0,\ x_0 + h,\ \ldots\ ,\ x_0 + nh$ с выдачей результатов на печать.

3. При условии, сформулированном в упражнении 1, составить алгоритм вычисления значений функции с запоминанием результатов в a-последовательность.

4. Составить алгоритм вычисления значений функции (7.14) для всех значений аргумента, заданных в виде a-последовательности

$${}'a = n;\ {}'(a \bigoplus i) = x_i\ (i = 1,\ 2,\ \ldots\ ,\ {}'a) \qquad (7.15)$$

с запоминанием результатов в b-последовательность.

5. Дана α-последовательность вида (7.15). Составить алгоритм нахождения суммы квадратов всех ее положительных членов.

6. Для последовательности (7.15) составить алгоритм вычисления разности и произведения суммы квадратов всех ее положительных членов на сумму квадратов всех отрицательных членов.

7. Для последовательности (7.15) найти первый член, не превышающий по модулю заданное число ε, и его адрес поместить по адресу s.

8. В примерах 1—7 рассмотреть схемы с применением фиксаторов, адресов сдвига, счетчиков, работающих по принципу сложения и вычитания. Рассмотреть варианты алгоритмов с применением формулы вычисляемого перехода (переключателя) и формулы циклирования.

9. Для последовательности (7.15) составить алгоритм определения:

а) разности между максимальным и минимальным ее членами;

б) большей из двух сумм всех ее членов, стоящих на четных и нечетных местах;

в) числа положительных ее членов;

г) числа перемен знака в последовательности;

д) суммы $\sum_{i=1}^{n-3} a_i \cdot a_{i+3}$.

10. Предполагая, что в последовательности (7.15) заданы целые числа, составить алгоритм определения:

а) суммы всех ее четных членов. Для определения четности числа воспользоваться константой, содержащей единицу в разряде, соответствующем в представлении целых чисел младшему разряду единиц, и нули в остальных разрядах;

б) адреса первого нечетного числа;

в) суммы первых ее четных членов, произведение которых не превосходит заданное число ε.

11. Составить алгоритм вычисления $n!$.

12. Составить алгоритм вычисления C_n^m.

13. Составить алгоритм вычисления с помощью ряда значений функции:

а) $\sin x$;

б) $\cos x$;

в) $\operatorname{arctg} x$.

14. Составить алгоритм одновременного вычисления с помощью ряда значений $\sin x$ и $\cos x$.

15. Составить алгоритм определения в множестве слов, заданных последовательностью их адресов (7.15), числа слов, совпадающих с заданным.

16. Предполагая, что в последовательности (7.15) заданы некоторые, быть может, совпадающие слова, составить алгоритм замены в ней всех слов, равных $'x$, словом $'y$.

В упражнениях 15 и 16 рассмотреть варианты программ с применением и без применения минус-штрих-операции.

17. Даны два вектора $X(x_i)$, $Y(y_i)$ размерности n последовательностями адресов вида (7.15), содержащими их компоненты.

Составить алгоритмы для вычисления:

а) косинуса угла между ними;

б) разности их модулей;

в) максимального из попарных произведений их компонент, т. е. из произведений вида $x_i \cdot y_i$;

г) числа перемен знака в последовательности попарных произведений их компонент $x_i \cdot y_i$;

д) числа попарно совпадающих их компонент $x_i = y_i$;

е) суммы произведений $\sum_{i=1}^{n-1} x_i \cdot y_{i-1}$.

18. Для примера 17 составить алгоритм:

а) построения вектора, четные компоненты которого равны компонентам вектора Y, а нечетные — компонентам вектора X;

б) замены четных компонент вектора X на нечетные компоненты вектора Y и наоборот (применить формулу обмена).

19. Для примера 17 составить алгоритм вычисления:

а) суммы тех компонент вектора Y, номера которых совпадают с номерами компонент вектора X, равных заданному числу $'a$;

б) замены нулями компонент вектора Y, указанных в а).

Составить вариант алгоритма с применением минус-штрих-операции. (Здесь вектор X можно рассматривать как вектор слов; ищутся слова, совпадающие со словом $'a$, и суммируются соответствующие компоненты вектора Y).

20. Для алгоритмов группового переноса, приведенных в примере (гл. 3 § 5), рассмотреть случай, когда последовательности адресов могут перекрываться.

21. Даны последовательностями адресов

$$\begin{aligned} &\alpha \oplus 1,\ \alpha \oplus 2,\ \ldots,\ \alpha + n; \\ &\beta \oplus 1,\ \beta \oplus 2,\ \ldots,\ \beta \oplus m, \end{aligned} \tag{7.16}$$

последовательности чисел a_i и b_j.

Составить алгоритм вычисления:

а) суммы всевозможных парных произведений $a_i \cdot b_i$;

б) суммы всех положительных парных произведений $a_i \cdot b_j$;

в) числа всех тех пар a_i, b_j, знаки которых совпадают;

г) суммы всех тех парных произведений $a_i \cdot b_j$, для которых $i \leqslant j$;

д) суммы всех тех положительных парных произведений $a_i \cdot b_j$, для которых $i > j$;

е) значений функции

$$f(i, j) = \frac{a_i \times b_j}{a_i^2 + b_j^2}$$

для всевозможных пар значений аргументов, для которых $i \leqslant j$.

Рассмотреть вариант алгоритма с выдачей результатов на печать и с запоминанием (в виде треугольной матрицы по строкам);

ж) то же, что и в е), но для функции

$$g(i, j) = \frac{a_i \times b_{j+3}}{a_i^2 + b_{j+3}^2}.$$

22. Предполагая, что последовательностями адресов (7.16) заданы некоторые слова, составить алгоритм:

а) вычисления числа тех слов α-последовательности, которые встречаются в β-последовательности;

б) определения максимального числа повторений слов последовательности α в последовательности β с запоминанием адреса слова, встречающегося максимальное число раз;

в) нахождения адреса первого слова α-последовательности, встречающегося в β-последовательности, и первого слова α-последовательности, встречающегося в β-последовательности;

г) запоминания адреса первого слова α-последовательности, встречающегося в β-последовательности, и адресов всех совпадающих с ним слов последовательности β (в s-последовательность).

Рассмотреть варианты алгоритмов с применением минус-штрих-операции.

23. Дана квадратная матрица $A(a_{ij})$ в виде α-последовательности адресов ее элементов (по строкам). Составить алгоритмы для определения:

а) суммы тех положительных ее элементов, для которых $i < j$;

б) номера столбца матрицы, сумма модулей элементов в котором максимальна;

в) следа матрицы;

г) транспонирования матрицы. Рассмотреть вариант алгоритма с применением формул обмена.

24. Составить алгоритм определения адреса первого элемента первого столбца матрицы A, встречающегося по меньшей мере один раз в остальных столбцах этой матрицы.

25. Составить алгоритм замены всех элементов матрицы A, совпадающих с $'a$, на $'b$; номера строк и столбцов этих элементов разместить в последовательности

$$J \oplus 1,\ J \oplus 2,\ \ldots;\ K \oplus 1,\ K \oplus 2,\ \ldots,\ K \oplus {}'K.$$

26. Составить алгоритм определения номера столбца матрицы, имеющего максимальное число совпадающих между собой элементов; элемент, встречающийся в столбцах максимальное число раз, поместить по адресу l.

Рассмотреть варианты алгоритмов в упражнениях 24—26 с применением и без применения минус-штрих-операции.

27. Рассмотреть задачи 23 а) — в) для случая симметричной матрицы, задаваемой по строкам в виде α-последовательности адресов ее диагональных и наддиагональных элементов.

28. Составить алгоритм вычисления квадрата матрицы A.

29. Составить алгоритм вычисления квадрата симметричной матрицы A, заданной своими диагональными и наддиагональными элементами в виде α-последовательности по строкам.

30. Составить алгоритм вычисления определителя симметричной матрицы, заданной, как и в примере 29.

31. Составить алгоритм определения номера того столбца симметричной матрицы, заданной, как и в примере 29, сумма модулей элементов которого максимальна.

ЛИТЕРАТУРА

1. Глушков В. М., Ющенко Е. Л., Вычислительная машина «Киев», Гостехиздат УССР, 1962.
2. Гнеденко Б. В., Королюк В. С., Ющенко Е. Л., Элементы программирования, Физматгиз, 1961.
3. Ершов А. П., Программирующая программа для быстродействующей электронной счетной машины, Изд-во АН СССР, 1958.
4. Домелки Б., Алгоритмы для распознавания последовательностей символов, Информационный бюллетень Венгерской АН, вып. 8, 1962.
5. Калужнин Л. А., Об алгоритмизации математических задач, «Проблемы кибернетики», вып. 2, Физматгиз, 1959.
6. Камынин С. С., Любимский Э. З., Шура-Бура М. Р., Об автоматизации программирования при помощи программирующей программы, «Проблемы кибернетики», вып. 1, Физматгиз, 1958.
7. Китов А. И., Криницкий Н. А., Электронные вычислительные машины и программирование, Физматгиз, 1959.
8. Колмогоров А. Н., Успенский В. А., К определению алгоритма, «Успехи математических наук», т. 13, 1948, № 4.
9. Королюк В. С., О понятии адресного алгоритма. «Проблемы кибернетики», вып. 4, Физматгиз, 1960.
10. Королюк В. С., Ющенко К. Л., Питання теорії та практики програмування, «Зб. ОЦ АН УРСР», 1961, № 1.
11. Ляпунов А. А., О логических схемах алгоритмов, Проблемы кибернетики, вып. 1, Физматгиз, 1958.
12. Марков А. А., Теория алгоритмов, «Труды Математического института АН СССР», вып. 42, 1954.
13. Рвачов В. Л., Про аналітичний опис деяких геометричних об'єктів, «Зб. праць ІК АН УРСР», 1963, № 1.
14. Система стандартных подпрограмм, Сборник под ред. М. Р. Шуры-Буры, Физматгиз, 1959.
15. Сообщение об алгоритмическом языке АЛГОЛ, Изд-во АН СССР, 1960.
16. Смольников Н. Я., Основы программирования для цифровой машины «Урал», изд-во «Советское радио», 1961.
17. Трахтенброт Б. А., Алгоритмы и машинное решение задач, Гостехиздат, 1957.

18. Фаддеев Д. К. и Фаддеев В. Н., Вычислительные методы линейной алгебры, Физматгиз, 1960.

19. Фридман В. М., Новые методы решения линейного операторного уравнения, ДАН СССР, 128, 1959, № 3.

20. Ющенко К. Л., Бистрова Л. П., Програмуюча програма, інформацією для якої служить адресний алгоритм, «Зб. ОЦ АН УРСР», 1961, № 3.

21. Ющенко К. Л., Адресні алгоритми та математичні машини, «Зб. ОЦ АН УРСР», 1961, № 2.

22. Ющенко К. Л., Рівні і стилі адресної мови та проблема автоматизації програмування, ДАН УРСР, 1962, № 6.

23. Ющенко К. Л., Адресні алгоритми та математичні машини ДАН УРСР, 1962, № 7.

24. Ющенко К. Л., Про повноту засобів адресної мови, ДАН УРСР, 1962, № 10.

25. Ющенко К. Л., Костюченко О. І., Алгоритм перекладу дужкового запису формул в бездужковий, «Зб. ОЦ АН УРСР», 1961, № 3.

26 Ющенко К. Л., Михайлова О. І., Алгоритм формальної перевірки правильності дужкового та бездужкового запису формул, «Зб. ОЦ АН УРСР», 1961, № 3.

ОГЛАВЛЕНИЕ

Глава V. **Адресный язык и метод универсальных программирующих программ**

Глава VI. **Построение схемы обозрения в кодах конкретных машин. (Машина «Урал»)**

Глава VII. **Примеры и упражнения**

Екатерина Логвиновна Ющенко
(канд. физ.-мат. наук)

Адресное программирование

Редактор издательства инж. *Л. Б. Навроцкая*
Переплет художника *С. М. Габовича*
Технический редактор *В. Н. Березовый*
Корректор *Н. П. Ефименко*

Сдано в набор 28.IV.63г. Подписано к печати 17.IX 1963 г. Формат бумаги 84×108/32. Объем: 9 физич. лист., 14,76, условн. лист., 15,7. учетно-издат. лист. Тираж 4000. БФ 03685. Цена 94 коп.

Государственное издательство технической литературы УССР
Киев, 4, Пушкинская, 28

Отпечатано с матриц Книжной фабрики им. Фрунзе Главполиграфиздата Министерства культуры УССР, Харьков, Донец-Захаржевская, 6/8, в типографии «Коммунист», Харьков, Пушкинская, 29. Зак. 0120.

94 коп.

Гостехиздат УССР
Киев-1963