ЦСУ СССР ГЛАВНОЕ УПРАВЛЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ РАБОТ

Н. Г. Бородулина, А. К. Зверева, Н. В. Жадан, Ю. П. Кирюхин, М. Я. Мазеев, Ф. Ф. Шиллер

СИСТЕМА АВТОМАТИЗАЦИИ ПРОГРАММИРОВАНИЯ АЛГЭМ

Под редакцией *доктора технических наук, профессора А. И. КИТОВА*

Статистика
Москва 1970

Система автоматизации программирования АЛГЭМ

Редактор *В. М. Мирзоева*
Техн. редактор *К. К. Сенчило*
Худ. редактор *Т. В. Стихно*

Сдано в набор 27/1II 1970 г. Подписано к печати 4/IX 1970 г. Формат бумаги 60X90$^{1}/_{16}$. Бумага № 2. Объем: 9,75 печ. л. Уч.-изд. л. 10,79.
Тираж 3900 экз. А07990. (Заказное издание Главмехсчета).

Издательство **«Статистика»,** Москва, ул. Кирова, 39. Заказ № 2585. Цена 1 р. 33 к.

Областная типография Ивановского управления по печати. г. Иваново, Типографская, 6.

СОДЕРЖАНИЕ

П Р Е Д И С Л О В И Е

Настоящая книга посвящена описанию системы автоматизации программирования экономических и математических задач для машины «Минск-22». Эта система основана на использовании алгоритмического языка АЛГЭМ. Слово АЛГЭМ является сокращенным названием данного языка, предназначенного для описания алгоритмов решения экономических и математических задач (**АЛГ**оритмы **Э**кономические и **М**атематические).

Система включает в себя входной язык, являющийся подмножеством языка АЛГЭМ, и транслятор, осуществляющий перевод программ с входного языка на машинный язык. Перевод производится в две стадии с использованием промежуточного языка.

Книга состоит из 3-х глав. В первой главе приводится формальное описание входного языка АЛГЭМ с использованием металингвистической символики, введенной Бэкусом для описания языка АЛГОЛ-60.

Во второй главе дается изложение языка АЛГЭМ и подробно рассматриваются особенности его реализации в описываемой системе автоматизации программирования. В частности, рассматриваются вопросы размещения транслятором значений числовых и строчных переменных, массивов, составных переменных и составных массивов в ячейках машины «Минск-22», вопросы подготовки исходных данных.

В третьей главе приводятся инструкции по работе с указанной системой автоматизации программирования на машине «Минск -22».

Для облегчения изучения данной книги и ускорения овладения описываемой системой автоматизации программирования полезно иметь первоначальное общее представление об основных чертах алгоритмического языка АЛГОЛ-60 (например, по первой части книги С. С. Лаврова «Универсальный язык программирования»). Кроме того, полезно познакомиться с общими характеристиками машины «Минск-22»> в частности с составом устройств машины, структурой ячейки памяти, системой команд.

Важнейшим элементом любой системы автоматизации программирования, с точки зрения пользователя, является входной язык этой системы.

Основным ядром этого языка служит сокращенный АЛГОЛ [1], дополненный элементами языка КОБОЛ [2]. По сравнению с языком АЛГОЛ-60 во входном языке АЛГЭМ произведены следующие сокращения:

1) отсутствуют условные выражения;
2) отсутствуют величины типа **логический** и логические выражения;
3) не допускается смешение внутри одного арифметического выражения переменных, чисел и указателей стандартных функций типа **целый** с переменными, числами и указателями стандартных функций типа **вещественный;**
4) в списке цикла допускается лишь один элемент. Этот элемент может представлять собой либо элемент типа арифметической прогрессии, либо список выражений;
5) в качестве условных операторов допускаются лишь операторы
если. <отношение> то <оператор>;
6) сокращен и упрощен раздел процедур. Допускается использование только процедур-кодов (с возможностью расширения их состава) и указателей стандартных функций;
7) в качестве границ в описаниях массивов можно использовать целые без знака, целые со знаком минус, а также простые переменные, имеющие по описанию тип **целый;**
8) переключательный описок содержит лишь метки;
9) в качестве меток могут использоваться только идентификаторы;
10) отсутствуют собственные величины.

Основное преимущество языка АЛГЭМ по сравнению с АЛГОЛОМ состоит в том, что язык АЛГЭМ позволяет эффективно оформлять алгоритмы экономических задач:

1. Наличие строк и переменных типа **строчный** дает возможность обрабатывать текстовую информацию, а также оформлять выдачу на АЦПУ-128 различной документации. Строки и переменные типа **строчный** могут использоваться в операторе присваивания (в том числе допускается преобразование значения типа **строчный** в значение типа **целый** или **вещественный** и наоборот), а также в качестве фактических параметров оператора процедуры-кода.

2. Указание вида для величин типа целый, **вещественный** или **строчный** позволяет редактировать значения, полученные в результате вычислений, перед выдачей их на узкую или широкую печать, а также дает возможность экономить оперативную память посредством упаковки таких отредактированных значений.

3. Описания составных величин (составных переменных и составных массивов) позволяют задавать структуры различных документов, обрабатывать эти документы, а также производить обмен соответствующей информацией между оперативной памятью и внешними устройствами.

4. Введение описаний внешних величин, располагаемых на магнитных лентах, дает возможность использовать любые лентопротяжные механизмы (ЛПМ) из числа имеющихся в составе оборудования ЭВМ «Минск-22».

5. Обеспечивается возможность использования специального оператора процедуры-кода, имеющего следующий вид: КОД ('<спецификация процедуры-кода>', <список фактических параметров>).

В качестве спецификации процедуры-кода могут использоваться как зарезервированные наименования, так и номера подпрограмм, включенных в библиотеку стандартных программ (БСП) пользователями в соответствии с инструкцией по включению в БСП новых процедур-кодов (см. разд. 3.10.2).

Список зарезервированных спецификаций процедур-кодов включает подпрограммы, обеспечивающие ввод числовой и текстовой информации с перфолент и перфокарт, вывод этой информации на перфоленты и

перфокарты, вывод на узкую и широкую печать, обмен с магнитными лентами.

6. В качестве средств, облегчающих отладку алгоритмов, записанных на языке АЛГЭМ, в язык включены два служебных оператора: оператор перехода по ключу, представляющий собой разновидность условного оператора, и оператор останова.

В связи с тем, что настоящая книга предназначена для пользователей системы автоматизации программирования, а не для разработчиков трансляторов и алгоритмических языков (хотя книга будет полезна и этим специалистам), в этой книге отсутствуют описания алгоритмов и программ транслятора и принципов его построения. Однако следует указать, что в системе АЛГЭМ применен синтаксический способ трансляции.

Основные принципы и алгоритмы работы синтаксических трансляторов разработаны и подробно описаны Айронсом и Ингерманом [7], [8].

В трансляторе АЛГЭМ-СТ-3 осуществлена реализация этих принципов применительно к языку АЛГЭМ и усовершенствованы основные алгоритмы по сравнению с описанными в литературе, а также разработан комплекс обслуживающих программ транслятора.

В трансляторе реализован разработанный Ф. Ф. Шиллер способ контроля программ на входном языке. Для этой цели множество синтаксических правил языка расширено таким образом, чтобы можно было анализировать не только правильные, но и ошибочные конструкции, а семантическая часть этих правил содержит указания о печати в выходной строке сведений о встретившихся синтаксических или семантических ошибках (см. [5]). Ф. Ф. Шиллер принадлежит также разработка всех синтаксических правил и входного языка.

Описываемый в данной книге транслятор АЛГЭМ-СТ-3 более эффективен в части сокращения времени трансляции и увеличения объема допускаемых к трансляции программ по сравнению с ранее опубликованным транслятором АЛГЭМ-СТ-2, получившим достаточно широкое применение. Это повышение эффективности достигнуто, в основном, за счет тщательной отработки Ю. П. Кирюхиным основных алгоритмов транслятора (блока ввода и кодировки, составителя таблиц, анализатора), в которых широко использованы оригинальные приемы размещения и обработки информации, в частности ассоциативные списковые структуры и магазины.

Система АЛГЭМ создавалась коллективом авторов настоящей книги под руководством редактора в период 1965—1969 гг. и в настоящее время полупила широкое применение. В процессе опытной эксплуатации системы, а затем практического ее использования были проверены и тщательно отработаны все алгоритмы и программы транслятора, выявлены и исправлены ошибки в синтаксических правилах и программах и произведена технологическая доработка системы, обеспечивающая расширение ее практических возможностей и удобство пользования.

К числу технологических особенностей системы АЛГЭМ относятся: наличие автоматического синтаксического контроля и печати ошибок, удобство внесения направлений в программу и исходные данные, широко развитая система контрольных остановов, гибкая система процедур-кодов, обеспечивающая включение в библиотеку стандартных программ любых новых специализированных подпрограмм, отвечающих нуждам пользователя, а также наличие комплекса обслуживающих программ транслятора (тест сохранности транслятора на МЛ, программа дублирования транслятора с МЛ на МЛ, программа проверки перфорации рабочих программ и др.).

Разработчики системы АЛГЭМ будут благодарны всем лицам, которые сообщат свои замечания как по самой системе автоматизации программирования, так и по данной книге по адресу: Москва, К-450, ул. Кирова, 39, изд-во «Статистика».

Доктор технических наук, профессор А. КИТОВ

ГЛАВА 1
ФОРМАЛЬНОЕ ОПИСАНИЕ ВХОДНОГО ЯЗЫКА АЛГЭМ

1.1. ФОРМАЛИЗМ ДЛЯ СИНТАКСИЧЕСКИХ ОПИСАНИЙ

Метаязык, используемый для описания языка АЛГЭМ, целиком взят из описания АЛГОЛа-60 [1].

1.2. ОСНОВНЫЕ СИМВОЛЫ, ДЕНТИФИКАТОРЫ, СТРОКИ, ОСНОВНЫЕ ПОНЯТИЯ

АЛГЭМ строится из следующих основных символов:

<основной символ> ::= <буква> | <цифра> | <ограничитель строки> |<ограничитель>

1.2.1. Буквы

<буква> ::= А|Б|В|Г|Д|Е|Ж|З|И|И|К|Л| М| Н|0|П|Р| С|Т|У|Ф|Х|Ц|Ч|Ш|Щ|Ь|Ы|Э|Ю|Я|D|F|G|I|J|L|N|Q|R|S|U|V|W|Z

Буквы не имеют индивидуального значения. Они используются для образования идентификаторов и строк (см. разд. 1.2.5 и 1.2.7).

1.2.2. Цифры

<цифра>::=0|1|2|3|4|5|6|7|8|9

Цифры используются для образования чисел, идентификаторов и строк.

1.2.3. Ограничители строк

< ограничитель строки >:: = ' | '

1.2.4. Ограничители

< ограничитель > : :=<знак операции> | <разделитель> | <скобка> | <описатель>
<знак операции>:: = <знак арифметической операции> | <знак операции отношения> | <знак операции следования>

<знак арифметической операции> : : = + | — | × | / | ↑
<знак операции отношения > :: = < |≤ | = |≥ | > | ≠
<знак операции следования> : : = **на** | **если** | **то** | **для** | **цикл** | **ко** | **стоп**
<разделитель> : : =,|.|10|:|;|: = |_| **шаг** |**до** | **примечание**
<скобка> : : = (|) | [|] | **начало** | **конец** | **составной** | **уровень**
<описатель > : : = **целый** | **вещественный** | **строчный** | **вид** | **массив** | **переключатель**

Ограничители имеют фиксированный смысл, который в большинстве случаев очевиден, а в остальных случаях будет указан позже.

Существуют следующие правила для включения текста примечаний между символами программы:

последовательность основных символов:	*эквивалентна*
; примечание <любая последовательность, не содержащая ; >;	;
начало примечание<любая последовательность не содержащая ;>;	**начало**
конец <любая последовательность, не содержащая **конец** или **; >**	**конец**

Эквивалентность здесь означает, что любая из трех конструкций, указанных в левой колонке, если она встречается вне некоторой строки, может заменяться соответствующим ей символом, указанным в правой колонке; эта замена не оказывает никакого влияния на работу программы. При этом считается, что конструкцию примечания, встретившуюся раньше при чтении текста слева направо, следует заменять прежде более поздних конструкций, содержащихся в этой последовательности.

1.2.5. Идентификаторы

1.2.5.1. Синтаксис

<идентификатор> :: = <буква> | <идентификатор> <буква> | <идентификатор> <цифра>

1.2.5.2. Примеры

А	Н17А
Bl	SIN
ТАБЕЛЬНЫЙ НОМЕР	А34КТМ

1.2.5.3. Семантика

Идентификаторы не имеют неизменно присущего им смысла, а служат только для обозначения простых переменных, массивов, составных переменных, составных массивов, меток и переключателей. Их можно выбирать произвольно, за исключением указанных в 1.4.7. Кроме того, идентификатор не может начинаться с буквы Я (см. 1.5.2 и 1.5.3).

Один и тот же идентификатор нельзя использовать для обозначения двух различных величин, за исключением следующих двух случаев:

а) если эти величины, согласно описаниям программы, имеют несовместные области действия (см. 1.5);

б) если эти величины, согласно описаниям программы, являются частями различных составных переменных или составных массивов (см. разд. 1.2.8, 1.3.1 да 1.5.3)

1.2.6. Числа

1.2.6.1. Синтаксис

<целое без знака>:: = <цифра>| <целое без знака> <цифра>
<целое>: :=<целое без знака> | + <целое без знака>| — <целое без знака>
<мантисса>: : = .<целое без знака>
<порядок> : : =$_{10}$<цифра>|$_{10}$+<цифра> | $_{10}$— <цифра>
<десятичное число> : : = <мантисса> | <целое без знака>|<мантисса>
<вещественное без знака> : : =<десятичное число> | <порядок>|<десятичное число> <порядок>
<число без знака> : : =<целое без знака>| <вещественное без знака>
<вещественное> : : =<вещественное без знака>| + <вещественное без знака> | — <вещественное без знака>
<число> : : = <целое> | <вещественное>

1.2.6.2. Примеры

0	-200.034
177.	$+07.43_{10}8$
.5384	$9.34_{10}+9$
+0.7300	$0.2_{10}-4$

1.2.6.3. Семантика

Десятичные числа имеют свой обычный смысл. Порядок — это масштабный множитель, выраженный как целая степень 10.

1.2.6.4. Типы

Целые числа имеют тип **целый**. Вещественные числа имеют тип **вещественный** (см. разд. 1.5.1).

1.2.7. Строки

1.2.7.1. Синтаксис

<строчный символ> :: =<буква>| <цифра>| <знак арифметической операции>|<знак операции отношения>| .|$_{10}$| _ /|*
<элемент строки> : : =<строчной символ>| <строка>
<открытая строка> : : = <элемент строки>| <открытая строка> | <элемент строки>
<строка> :: ='<открытая строка>'

1.2.7.2. Примеры

_ ОКТЯБРЯ _ 1967 _ Г.'
'125-'ABC+'A+B'

1.2.7.3. Семантика

Строки имеют тип **строчный** (см. разд. 1.5.1). Они введены для того, чтобы дать возможность оперировать в языке последовательностями основных символов. Символ __ означает пробел и не имеет смысла вне строк.

Открытые строки могут использоваться как значения (см. разд. 1.5.1) переменных, имеющих по описанию тип **строчный.**

1.2.8. Величины, классы и области действия

Различаются следующие классы величин: простые переменные, массивы, составные переменные, составные массивы, метки и переключатели.

Область действия величины—это совокупность операторов и выражений, внутри которой имеет силу описание идентификатора, связанного с этой величиной. (О метках см. разд. 1.4.1.3.**)**.

1.2.9. Значения и типы

Значение — это некоторое упорядоченное множество, элементами которого могут быть числа и открытые строки либо метка. Это может быть одно число, одна строка или совокупность чисел, совокупность открытых строк.

О некоторых синтаксических единицах говорится, что они обладают значениями. Во время выполнения программы эти значения могут изменяться. Значения выражений и их компонентов определяются в разд. 1.3. Значение идентификатора массива есть упорядоченное множество значений соответствующего массива переменных с индексами (см. разд. 1.3.1. и 1.4.1). Значение идентификатора составной переменной есть упорядоченное множество значений элементарных величин, входящих в ее состав (см. разд. 1.3.1); значение идентификатора составного массива есть упорядоченное множество значений соответствующего массива составных переменных с индексами (см. разд. 1.3.1).

Различные типы **(целый, вещественный, строчный)** обозначают свойства значений. Они характеризуют значения элементарных величин (см. разд. 1.3.1). Специальным типом является пара скобок **составной** и **уровень.** Он обозначает единицу информации, содержащую связанные между собой единицы информации, возможно, с различными свойствами (типами) значений.

* Используется как вертикальный разделитель при выдаче на АЦПУ.

1.3. ВЫРАЖЕНИЯ

В данном языке первичными компонентами программ, описывающих алгоритмические процессы, являются арифметические, строчные и именующие выражения, а также отношения. Компонентами этих выражений, помимо некоторых ограничителей, являются числа, строки, переменные, указатели стандартных функций и элементарные арифметические операции, а также некоторые операции отношения. Поскольку синтаксическое определение как переменных, так и указателей стандартных функций содержит выражения, определение выражений и их компонентов непременно должно быть рекурсивным.

<выражение> : : =<арифметическое выражение> | <строчное выражение>| <отношение>| <именующее выражение>

1.3.1. Переменные

1.3.1.1. Синтаксис

<идентификатор переменной > : : = <идентификатор>
<простая переменная> : : = <идентификатор переменной>
<индексное выражение> : : = <арифметическое выражение>
<список индексов>:: = <индексное выражение> | <список индексов>, <индексное выражение>
<идентификатор массива> : : =<идентификатор>
<переменная с индексами> : : = <идентификатор массива>[<список индексов>]
<элементарная переменная> : : = <простая переменная> |<переменная с индексами>
<идентификатор составной переменной> : : = <идентификатор>
<идентификатор составного массива> :: = <идентификатор>
<уточнение>:: = .<идентификатор составной переменной>|·<идентификатор составного массива>[<список индексов>]
<переменная> :: =<элементарная переменная>|<переменная>,<уточнение>

1.3.1.2. Примеры

A17
H[7,2]
XУ [sin (H↑P1/2.0), РТЗ, Н, 4]
A.B1 [2, IK]. P7

1.3.1.3. Семантика

Переменная—это наименование, данное некоторому отдельному значению. Это значение можно использовать в выражениях для образования других значений, а также можно изменять посредством оператора присваивания (см. разд. 1.4.2). Тип значения конкретной переменной определяется описанием этой переменной (см. разд. 1.5.1) или описанием соответствующего идентификатора массива (см. разд. 1.5.2 и 1.5.3).

1.3.1.4. Индексы

1.3.1.4.1. Переменные с индексами представляют собой значения, которые являются компонентами многомерных массивов (см. разд. 1.5.2 и 1.5.3). Каждое арифметическое выражение из списка индексов занимает одну индексную позицию переменной с индексами и называется индексом. Полный список индексов заключается в индексные скобки []. Какой именно компонент массива упоминается с помощью переменной с индексами, определяется по фактическому числовому значению его индексов (см. разд. 1.3.3).

1.3.1.4.2. Каждая индексная позиция выступает как переменная типа **целый,** и вычисление индекса понимается как присваивание значения этой фиктивной переменной (см. разд. 1.4.2.4). Значение переменной с индексами определено только в том случае, когда значение индексного выражения находится в пределах границ индексов массива (см. разд. 1.5.2).

1.3.1.5. Уточнения

Уточнения (см. разд. 1.2,5.3 и 1.5,3) используются для исключения двусмысленностей, связанных с использованием одного и того же идентификатора для обозначения переменных, имеющих по описанию пересекающиеся области действия. Уточнения представляют собой ссылку на более крупные единицы информации, в состав которых (согласно описанию) входит данная переменная, и отделяются друг от друга и от элементарной переменной, которую они поясняют, точками. Выписывать следует все уточнения в порядке их иерархии.

1.3.2. Указатели стандартных функций

1.3.2.1. Синтаксис

<идентификатор стандартной функции> :: = **abs**|**знак sqrt**|**sin**|**cos**| **arctg**|**exp**|**ln**
<указатель стандартной функции> : : = <идентификатор стандартной функции> (<арифметическое выражение>)

1.3.2.2. Примеры

abs (A-B)
ln(P + Q↑H)

1.3.2.3. Семантика

Указатели функций определяют отдельные числовые значения. Функции **abs** (А) и **знак** (А) определены для А, являющегося как вещественным, так и целым выражением. Остальные функции определены только в случае вещественного выражения. Указатель функции **знак** (А) определяет значение типа **целый.** Указатель функции

abs (A) определяет значение, тип которого совпадает с типом значения выражения A. **Все** остальные функции дают значения типа **вещественный.**

1.3.3. Арифметические выражения

1.3.3.1. Синтаксис

<арифметическое выражение> : : = <целое выражение> | <вещественное выражение >

1.3.3.2. Семантика

Арифметическое выражение задает правило для вычисления числового значения. Различаются два вида арифметических выражений: целые выражения, определяющие значения типа **целый,** и вещественные выражения, определяющие значения типа **вещественный.**

1.3.4. Целые выражения

1.3.4.1. Синтаксис

<первичное целое выражение> : : = <целое без знака> | <переменная> | < указатель стандартной функции>| (<целое выражение>)

<целый множитель> : : = <первичное целое выражение> | <первичное целое выражение>↑<целое без знака>
<целый одночлен> : : =<целый множитель> | <целый одночлен>×<целый множитель >
<операция типа сложения>: : = + | —
<целое выражение> : : =<целый одночлен>| <операция типа сложения> <целый одночлен> | <целое выражение> <операция типа сложения> <целый одночлен>

1.3.4.2. Примеры

Первичные целые выражения:

N знак (A—X+P)
K2 [H + P[I, J]] (I + 2×P-K)

Целые множители:

N 3
(I+2×P-K)↑3

Целые одночлены: K2 [2+B [I, J]] × (1+2×P—K)↑3
знак (A-X+P)×B×5×H [J]. K2.K [L]

Целое выражение:

-N3+K2 [-2+B [I, J]] × (1+2×P—K)↑3-знак (A) ×N×5

1.3.4.3. Семантика

Значение целого выражения определяется в результате выполнения указанных в нем арифметических операций над значениями первичных целых выражений. Подробно это объяснено в разделах 1.3.4.4 и 1.3.4.5. Что понимается под значением первичного целого выражения, становится ясно, когда этим выражением служит целое число без знака.

Для переменной значением является ее текущее (последнее присвоенное ей) значение, а для указателей стандартных функций — значение, полученное с помощью соответствующей стандартной программы. Наконец, для целого выражения, заключенного в скобки, значение должно быть найдено путем рекурсивного процесса, сводящего вычисление этого значения к вычислению значений первичных целых выражений остальных трех видов.

1.3.4.4. Операции и типы

Компоненты целых выражений (за исключением аргументов стандартных функций и индексных выражений) должны быть по описанию типа **целый.** Смысл операций, встречающихся в целом выражении, определяется следующими правилами:

1.3.4.4.1. Знаки операций „ + ", „—" и „×" означают соответствующие действия (сложение, вычитание, умножение).

1.3.4.4.2. Знак операции деление „/" в целом выражении не допускается (см. 1.4.7).

1.3.4.4.3. Знак операции „↑" в выражении <первичное целое выражение>↑<целое без знака> означает действие возведения в степень, где <первичное целое выражение> служит основанием, а<целое без знака> — показателем степени. Согласно синтаксису данного языка, повторное возведение целого множителя в степень не допускается, и показателем степени может быть только целое число без знака.

1.3.4.5. Старшинство операций

Операции в целом выражении выполняются с учетом следующих правил:

1.3.4.5.1. Согласно синтаксису, данному в разделе 1.3.4.1, выдерживается следующая очередность выполнения операций:

↑
×
+ —

Операции одной очереди выполняются в порядке их появления в выражении слева направо.

Желаемый порядок выполнения операций всегда может быть достигнут соответствующей расстановкой скобок.

1.3.5. Вещественные выражения

1.3.5.1. Синтаксис

<первичное вещественное выражение> :: = <вещественное без знака>| <переменная>| <указатель стандартной функции>|(<вещественное выражение>)
<операция типа умножения> : : =×|/
<вещественный множитель> : : =<первичное вещественное выражение>|<вещественный множитель>↑<первичное вещественное выражение>
<вещественный одночлен> : : =<вещественный множитель > | <вещественный одночлен> <операция типа умножения> <вещественный множитель >
<вещественное выражение> : : = <вещественный одночлен> | <операция типа сложения> <вещественный одночлен>| <вещественное выражение> < операция типа сложения> <вещественный одночлен>

1.3.5.2. Примеры

7.394_{10}-8
SUM
W [1+2,8]
cos (Y+Z×3.3)
(A+3.0/Y-VU↑2.0)
P. H [J, K] ↑**cos** (Y+Z×3.0)
—7.394_{10}—8↑W [I+2,8]↑ (A+3.0)- PK×B1.B2 [H, P]
OMEGA×SUM↑**cos** (Y—Z×3.0)/7:394_{10}—8↑W [I+2,8]↑ (A+3.0/Y—V×V)

1.3.5.3. Семантика

Вещественное выражение является правилом для вычисления числового значения типа вещественный. Принципы вычисления значения аналогичны принципам, изложенным в разделе 1.3.4 для целых выражений.

1.3.5.4. Типы

Переменные и указатели функций, используемые в качестве первичных вещественных выражений, должны иметь тип **вещественный** (ем. разделы **1.5.1** и 1.5.2).

1.3.5.5. Операции

Знаки операций имеют прежний смысл со следующими дополнениями:

1.3.5.5.1. Знак операции „/" в выражении <вещественный одночлен> / <вещественный множитель> означает деление значения одночлена на значение множителя. Это выражение не определено, если значение множителя равно нулю.

1.3.5.5.2. Знак операции „↑" в выражении <вещественный множитель> ↑ <первичное вещественное выражение> означает возведение в степень. Если А обозначает значение множителя ('основания степени), а В — значение первичного выражения (показателя степени), то результат равен *exp* (B×*ln* (A)) в случае, если A>0, и не определен, если A≤0.

1.3.5.6. Старшинство операций

Правила старшинства, приведенные в разделе 1.3.4.5, сохраняются с учетом того, что операции, обозначаемые знаками „×" и „/", стоят на одной очереди. Так, например, выражение

2.0↑M↑A означает (2.0↑M) ↑ A, а выражение
A/B×7.0/(P-Q)×V/S означает ((((A/B) ×7.0)/(P-Q)) ×V)/**S**

1.3.5.7. Точность представления вещественных величин

Числа и переменные типа вещественный должны рассматриваться как объекты, определенные с конечной точностью. Аналогично, при вычислении вещественных выражений подразумевается возможность отклонения от математически определенного результата, и, следовательно, допускается, что одно конкретное представление может привести к значению вещественного выражения, отличному от того, которое будет получено в каком-либо другом конкретном представлении. Контроль точности вычислений, если это необходимо, должен производиться методами численного анализа. Этот контроль должен рассматриваться как часть описываемого вычислительного процесса и, следовательно, выражаться в терминах алгоритмического языка.

Одним из способов задания точности вычислений является использование указания вида числа (см. разд. 1.5,1).

1.3.6. Строчные выражения

1.3.6.1. Синтаксис

<строчное выражение> : : =<строка> | <переменная>

1.3.6.2. Примеры

A1	'ИВАНОВ'
P2.K13[J1,K]	СТАВКА

1.3.6.3. Семантика

Строчное выражение является правилом для вычисления открытой строки.

1.3.6.4. Типы

Если строчным выражением является переменная, то она должна по описанию иметь тип **строчный**.

1.3.7. Отношения

1.3.7.1. Синтаксис

<отношение>.: : =<целое выражение> знак операции отношения> <целое выражение>|<вещественное выражение> <знак операции отношения><вещественное выражение>

1.3.7.2. Примеры

I ≠0	X≥0.0
M [J-1]<N	A+B >Z—2.0/D↑Q

1.3.7.3. Семантика

Отношение — выражение, которое может быть истинно или ложно в зависимости от того, выполняется или нет зависимость между значениями входящих в него арифметических выражений. Арифметические выражения быть должны одного типа. Истинность отношения определяется после вычисления значений арифметических выражений. При использовании отношений, содержащих вещественные выражения, следует считаться с возможным различием значения истинности отношения в данном конкретном представлении и других конкретных представлениях из-за неточности значений вещественных выражений (см. разд. 1.3.5.7).

1.3.8. Именующее выражение

1.3.8.1. Синтаксис

<метка> : : =<идентификатор>
<идентификатор переключателя>:: = <идентификатор>
<указатель переключателя> :: = <идентификатор переключателя> [< индексное выражение>]
<именующее выражение> : : =<метка> <указатель переключателя>

1.3.8.2. Примеры

Р9
М.12
КЛЮЧ [I]

1.3.8.3. Семантика

Именующее выражение является правилам для получения метки оператора (см. разд. 1.4). Если именующее выражение является меткой, то требуемый результат получен. Указатель переключателя отсылает к соответствующему описанию переключателя (см. разд. 5.4) и по фактическому числовому значению своего индексного выражения выбирает одну из меток, перечисленных в описании переключателя, отсчитывая эти метки слева направо.

1.3.8.4. Индексное выражение

Вычисление значения индексного выражения в указателе переключателя аналогично такому же вычислению для переменных с индексами (см. разд. 1.3.1.4.2). Значение указателя переключателя определено только в том случае, когда индексное выражение принимает одно из положительных значений 1,2,..., *n,* где *n* — число членов в переключательном списке.

1.4. ОПЕРАТОРЫ

Единицы действий в языке называются операторами. Обычно они выполняются в той последовательности, в какой написаны. Однако эта последовательность действий может прерываться операторами перехода, которые явно определяют своего преемника, и сокращаться условными операторами, которые могут вызывать пропуск некоторых операторов.

Для того чтобы имелась возможность указывать фактический порядок следования операторов в процессе работы, операторы могут снабжаться метками.

|Ввиду того, что последовательности операторов могут группироваться в составные операторы и блоки, определение оператора должно быть рекурсивным. Кроме того, поскольку описания, о которых говорится в разд. 1,5, существенно входят в эту синтаксическую структуру, синтаксическое описание оператора должно предполагать, что описания уже определены.

1.4.1. Составные операторы и блоки

1.4.1.1. Синтаксис

<основной оператор> : : =<оператор присваивания>|<оператор перехода>| <пустой оператор>| <оператор процедуры-кода>
<оператор> :: — <основной оператор>| <условный оператор**>** | <оператор цикла> | <составной оператор> | <блок>
<конец составного>:: = **<**оператор**>конец** | <оператор>;<конец составного>
< метка > : : =<идентификатор>
<составной оператор> : : = **начало** < конец составного>|<метка> : <составной оператор>
<начало блока>:: **= начало** <описание> | <начало блока>;<описание>
<блок> : : =<начало блока>; <конец составного> | <метка> : <блок>
<программа> : : =<блок>

Этот синтаксис можно проиллюстрировать следующим образом. Обозначим произвольные операторы, описания и метки буквами S, D и L соответственно. Тогда основные синтаксические единицы примут следующий вид:

Составной оператор:

L:L:L:...L**: начало** S;S;...; S **конец**

Блок:

L:L:...L: **начало** D;D;...; D; S;...; S **конец**

При этом надо помнить, что каждый из операторов может в свою очередь быть составным оператором или блоком.

1.4.1.2. Примеры

Основные операторы:

A:=P+Q
на СТАВКА
НАЧАЛО: ПРОДОЛЖЕНИЕ**:** W**:** = 7.993

Составной оператор:

начало X:=0.0;
для K: = 1 **шаг** 1 **до** P **цикл**
X:=X+A[1]×A[1];
если X>Q **то на** ОСТАНОВ;
AB:T2:W:=X+B0B
конец

Блок:

Q: **начало целый** I, K; **вещественный** W;
для 1:=1 **шаг** 1 **до** M **цикл**
для K: =1+1 **шаг** 1 **до** M **цикл**
начало W: = A [I , K] ; A [I, K]: = A [K, I];
A [K, I]:=W
конец ПО I, ПО K
конец БЛОКА Q

1.4.1.3. Семантика

Каждый блок автоматически вводит новый уровень обозначений. Это реализуется следующим образом. Любой идентификатор, встречающийся внутри блока, можно определить посредством соответствующего описания как локальный в данном блоке (см. разд. 1.5). Это означает, во-первых, что объект, представленный этим идентификатором внутри данного блока, не существует вне блока и, во-вторых, что любой объект, представленный тем же идентификатором вне данного блока, нельзя использовать внутри блока.

Идентификаторы (за исключением тех, которые изображают метки), встречающиеся внутри блока и не описанные в нем, не. локализуются в блоке, т. е. представляют одни и те же объекты как внутри блока, так и в соседних блоках. Метка, отделенная от оператора двоеточием, т. е. помечающая этот оператор действует так, как будто она описана в заголовке наименьшего объемлющего блока, т. е. наименьшего блока, чьи скобки **начало** и **конец** заключают этот оператор.

Так как оператор в блоке может в свою очередь тоже быть блоком, концепции локальности и нелокальности в блоке следует понимать рекурсивно.

1.4.2. Операторы присваивания

1.4.2.1. Синтаксис

<левая часть> : : = <переменная> : =
<список левой части> : : =<левая часть> | <список левой части> <левая часть>
<оператор присваивания> : : =<список левой части> <арифметическое выражение> | <список левой части> <строчное выражение>|
<метка>:
<оператор присваивания>

1.4.2.2. Примеры

H: = P[0] :=B1 :=B1 + 1.0+H
N:=N+1
I:=K× (K+I)↑2-L
P [I, K+2]. B2: =3.0-**arctg** (P15×ZETA)
Bl : ='ВЕДОМОСТЬ'

1.4.2.3. Семантика

Операторы присваивания служат для присваивания значения выражения (арифметического или строчного) одной или нескольким переменным. Этот процесс происходит в общем случае в три этапа.

1.4.2.3.1. Значения всех индексных выражений, встречающихся в переменных левой части, вычисляются в порядке слева направо.

1.4.2.3.2. Вычисляется значение выражения в операторе.

1.4.2.2.3.3. Значение выражения присваивается всем переменным списка левой части, при этом индексные выражения имеют значения, вычисленные на шаге 1.4.2.3.1.

Все переменные списка левой части должны по описанию иметь один и тот же тип. Если тип выражения отличается от типа переменных списка левой части, то считается, что в конце шага 1.4.2.3.2 автоматически включается соответствующая функция преобразования. Имеется в виду, что для преобразования типа **вещественный** в тип **целый** функция преобразования выдает результат, эквивалентный

entier (A+0.5),

где A — значение выражения.

Если переменные списка левой части имеют по описанию тип **целый** или **вещественный**, а выражение является арифметическим, то в том случае, если все или некоторые переменные описка левой части имеют указания вида, присваивание производится в соответствии с этими указаниями вида, т. е. с редактированием (см. разд. 1.5.1).

Если переменные описка левой части имеют по описанию тип строчный, а выражение является арифметическим, то в качестве этого выражения может выступать лишь переменная, имеющая согласно описанию указание вида (см. разд. 1.5.1).

Если переменные списка левой части имеют по описанию тип **целый** или **вещественный**, а выражение является строчным, то в качестве этого выражения может выступать строчная переменная, а переменные описка левой части должны иметь согласно описанию указания вида (см. разд. 1.5.1).

Следует иметь в виду, что в последних двух случаях присваивание производится символ за символом (в соответствии с указанием вида принимающей переменной), и пересчет символов идет слева направо.

1.4.3. Операторы перехода

1.4.3.1. Синтаксис

<оператор перехода>:: =**на** <именующее выражение>

1.4.3.2. Примеры

на М3
на ВЫХОД [J+I]

1.4.3.3. Семантика

Оператор перехода прерывает естественную последовательность действий, задаваемую порядком написания операторов, явно определяя своего преемника по значению именующего выражения. Таким образом, следующим выполняемым оператором будет тот, который имеет это значение в качестве своей метки.

1.4.3.4. Ограничения

Так как метки всегда локальны, то ни один оператор перехода не может извне вести к метке внутри блока. Оператор перехода может, однако, вести извне к метке внутри составного оператора.

1.4.3.5. Переход при неопределенном указателе переключателя

В том случае, когда именующее выражение есть указатель переключателя, значение которого не определено, оператор перехода также не определен.

1.4.4. Пустые операторы

1.4.4.1. Синтаксис

<пустой оператор > : : =<метка> : | <метка> : <пустой оператор>

1.4.4.2. Примеры

M:
начало ...; ВЫХОД: **конец**

1.4.4.3. Семантика

Пустой оператор не выполняет никакого действия. Он служит для помещения метки.

1.4.5. Условные операторы

1.4.5.1. Синтаксис

<переход по ключу> :: **=ко** <целое без знака> **на** <метка>
<условие>: : ==**если**<отношение>**то**
<условный оператор>:: = <условие><оператор>|<переход по ключу>|<метка> : <условный оператор>

1.4.5.2. Примеры

если X>0 то на P2
если K≠J **то** M1:A:=A+1
ко 2 на M3

1.4.5.3. Семантика

Условные операторы приводят к пропуску или выполнению некоторых операторов в зависимости от того, истинно или ложно значение отношения в условии.

Оператор, входящий в условный оператор, выполняется в том случае, когда отношение, входящее в условие, является истинным. В противном случае он пропускается, и действия продолжаются, начиная со следующего оператора.

Переход по ключу вызывает выполнение оператора перехода, входящего в синтаксическую конструкцию этой разновидности условного оператора, в том случае, когда ключ, номер которого задается целым без знака, отключен (ко). В противном случае оператор перехода по ключу эквивалентен пустому оператору.

1.4.6. Операторы цикла

1.4.6.1. Синтаксис

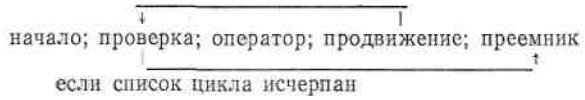
«параметр цикла> : : =<простая переменная>
<список выражений > : : =<арифметическое выражение> | <список выражений >,<арифметическое выражение >
<арифметическая прогрессия> : : =<арифметическое выражение> **шаг**<арифметическое выражение>**до**<арифметическое выражение>
<список цикла> : : =<список выражений> | < арифметическая прогрессия>
<заголовок цикла> :: **=для** <параметр цикла> : =список цикла**>цикл**
<оператор цикла> : : = <заголовок цикла> <оператор>|<метка> : <оператор цикла>

1.4.6.2. Примеры

```
для P:=1 шаг 1 до M цикл A [P] : =B [P]
для X : =0.1, 0.2, 0.3, 0.5, 0.8, 1.0, 1.5 цикл
        начало для I : =—M шаг 1 до M цикл
          A [ I ] : = l n ( X ) ;
          КОД ('ПЧ_2-10',A)
        конец
```

1.4.6.3. Семантика

Заголовок цикла заставляет стоящий за ним оператор S повторно выполняться нуль или более раз. Кроме того, он осуществляет последовательные присваивания значений переменной, служащей параметром цикла. Этот процесс может быть пояснен следующей схемой:



В этой схеме слово «начало» означает: произвести первое присваивание в заголовке цикла; «продвижение» означает: произвести следующее присваивание в заголовке цикла; «проверка» определяет, 'было ли сделано последнее присваивание. Если оно было сделано, то выполнение продолжается с преемника оператора цикла. В противном случае выполняется оператор, стоящий за заголовком цикла.

1.4.6.4. Параметр цикла

В качестве параметра цикла допускается простая переменная, имеющая согласно описанию тип **целый** или **вещественный** и не имеющая указания вида (см. разд. 1.5.1).

1.4.6.5. Список цикла

Список цикла дает правило для получения значений, которые последовательно присваиваются параметру цикла.

1.4.6.5.1. Список выражений. Последовательность значений, присваиваемых параметру цикла этим списком, получается в результате поочередного вычисления значений выражений, указанных в списке. Значение каждого выражения вычисляется непосредственно перед соответствующим выполнением оператора S.

1.4.6.5.2. Арифметическая прогрессия. Процесс выполнения, заданный оператором цикла вида

для V: = A **шаг** B до C **цикл**

может быть описан средствами языка АЛГЭМ следующим образом (при условии, что B1 и C1 того же типа, что и V):

```
V:=A;   B1:=B;   Ml : C1 : =C;
если знак ((V-C1) ×B1)>0 то на M2; S;
B1:=B;   V:=V+B1;   на Ml; M2 :...
```

1.4.6.6. Значение параметра цикла после выхода

После выхода из оператора S (если предполагается, что он является составным) с помощью какого-либо' оператора перехода значение параметра цикла будет таким, каким оно было непосредственно перед выполнением оператора перехода.

Если, с другой стороны, выход из цикла вызван исчерпанием списка цикла, значение параметра цикла не определено.

1.4.6.7. Оператор перехода, ведущий в оператор цикла

Результат действия оператора перехода, стоящего вне оператора цикла и обращающегося к метке внутри оператора цикла, не определен.

1.4.7. Операторы процедур-кодов

1.4.7.1. Синтаксис

<групповой параметр>::=<идентификатор массива> | <идентификатор составной переменной > | <идентификатор составного массива> [<список индексов>]|<идентификатор составного массива>|<групповой параметр> <уточнение> | <идентификатор внешнего массива>|< идентификатор внешней составной >|<идентификатор внешнего составного массива>[<список индексов>] | <идентификатор внешнего составного массива >
<фактический параметр>::=<число без знака> | <строка> | <переменная>| <групповой параметр>
<<восьмеричная цифра> : : =0| 1[2|3|4|5|6[7
<адрес> :: = <восьмеричная цифра> ... <восьмеричная цифра>

8 раз

<номер подпрограммы > : : =<адрес>
<спецификация процедуры-кода>::='2-10'|'10-2'|'ПЧ'|'ПЧ_10'|ПЧ_2-10'|'ПЧ_АЦПУ'|'ПРЛ'|'ПРЛ_10'|'ПРЛ_2-10'|'ПРЛ_С'|'ПРК'|'ПРК_ 10'|'ПРК_ 2-10'|'ПРК_ С'|'ВВОДЛ'|'ВВОДЛ_10-2'|'ВВОДЛ_ С'|'ВВОДК'|'ВВОДК _ 10'|'ВВОДК_ 10-2'|'ВВОДК_ С'|'Д'| 'ЗАПИСЬ' | 'ЧТЕНИЕ' | 'ВЫХОД' | '<номер подпрограммы>'
<список фактических параметров>:: = <спецификация процедуры-кода> | <список фактических параметров>,<фактический параметр>
<оператор процедуры-кода> :: —КОД (<список фактических параметров>)

1.4.7.2. Примеры

КОД ('ВВОДЛ_ (10-2', А, В [J], С)
КОД ('Д', А1, В1, А2, В2)
КОД ('ПРК_ 10', А, В)
КОД ('ВЫХОД', 2)
КОД ('00000161', 3, К, L, М)
КОД ('ЗАПИСЬ', А, Я03006000, В1, Я03004000 [К])
КОД ('ЧТЕНИЕ', Я01004000 [J], Р)

1.4.7.3. Семантика

Оператор процедуры-кода вызывает обращение к стандартной программе, записанной в кодах ЭВМ «Минск-22».

1.4.7.3.1. Фактические параметры. Для спецификаций с номерами 1—21 (см. таблицу спецификаций процедур-кодов) число фактических параметров может быть любым. Для спецификаций с номерами 23 и 24 в качестве фактических параметров можно использовать только групповые параметры. Их число должно быть четным. В каждой паре аргументов первым всегда записывается аргумент-отправитель, а вторым — аргумент-получатель.

Оператор процедуры-кода со спецификацией с номером 25 имеет лишь один/фактический параметр—целое без, знака или простую переменную типа **целый** без указания вида. Такой оператор процедуры-кода используется для сегментации программ, составляемых СТ-3.

О процедурах-кодах, спецификациями которых служат номера программ, см. разд. 3.10.2.

1.4.8. Операторы останова

1.4.8.1. Синтаксис

<оператор останова> :: =**стоп**<целое без знака>

1.4.8.2. Примеры

стоп 1
стоп 25

1.4.8.3. Семантика

Оператор останова вызывает останов рабочей программы с высвечиванием на сумматоре (в двоичном коде) целого без знака (≤4095), использованного в конкретном операторе останова, и адреса команды, соответствующей данному оператору. При нажатии кнопки «пуск» выполнение программы продолжится со следующего оператора.

Таблица спецификаций процедур-кодов

	Спецификация	Выполняемое действие
1	'2—10'	Перевод чисел из двоичной системы в десятичную
2	'10—2'	Перевод чисел из десятичной системы в двоичную
3	'ПЧ'	Печать на ТБПМ чисел в восьмеричной системе
4	'ПЧ 10'	Печать на ТБПМ чисел в десятичной системе
5	'ПЧ_2—10'	Печать на ТБПМ чисел с предварительным переводом из двоичной системы в десятичную
6	'ПЧ АЦПУ	Печать на АЦПУ значений строчных величин

	Спецификация	Выполняемое действие
7	'ПРЛ'	Перфорация на перфоленту чисел в восьмеричной системе
8	'ПРЛ_10'	Перфорация на перфоленту чисел в десятичной системе
9	'ПРЛ_2—10'	Перфорация на перфоленту чисел с предварительным переводом из двоичной системы в десятичную
10	'ПРЛ_С	Перфорация на перфоленту значений строчных величин
11	'ПРК'	Перфорация на перфокарты чисел в восьмеричной системе
12	'ПРК_10'	Перфорация на перфокарты чисел в десятичной системе
13	'ПРК_2—10'	Перфорация на перфокарты чисел с предварительным переводом из двоичной системы в десятичную
14	'ПРК_С'	Перфорация на перфокарты значений строчных величин
15	'ВВОДЛ'	Ввод с перфоленты чисел
16	'ВВОДЛ_ 10—2'	Ввод с перфоленты чисел с последующим переводом из десятичной системы в двоичную
17	'ВВОДЛ_С'	Ввод с перфоленты значений строчных величин
18	'ВВОДК'	Ввод с перфокарт чисел в восьмеричной системе
19	'ВВОДК_10-2'	Ввод с перфокарт чисел в десятичной системе
20	'ВВОДК_10-2'	Ввод с перфокарт чисел с последующим переводом из десятичной системы в двоичную
21	'ВВОДК_С'	Ввод с перфокарт значений строчных величин
22	'Д'	Подпрограмма деления целых чисел с нахождением частного и остатка; первый аргумент — делимое, второй аргумент — делитель, третий аргумент — частное,
23	'ЗАПИСЬ'	Запись внутренней величины (заданной идентификатором) по адресу, указанному идентификатором соответствующей внешней величины
.24	'ЧТЕНИЕ'	Чтение внешней величины
25	'ВЫХОД'	Появление в программе оператора процедуры-кода с этой спецификацией означает конец выполнения сегмента программы с одновременной передачей управления сегменту, номер которого указан фактическим параметром этого оператора

1.5. ОПИСАНИЯ

Описания служат для того, чтобы определить некоторые свойства величин, используемых в программе, и связать эти величины с идентификаторами. Описание идентификатора имеет силу только в одном блоке. Вне этого блока тот же идентификатор можно использовать для других целей (см. раздел 1.4.1.3).

В процессе работы программы это влечет за собой следующее: с момента входа в блок (через начало, так как внутренние метки локальны и, следовательно, недостижимы извне) все идентификаторы, описанные в блоке, приобретают смысл, вытекающий из природы данных описаний. Если эти идентификаторы уже были определены другими описаниями, находящимися вне блока, то на некоторое время они получают новый смысл. С другой стороны, те идентификаторы, которые не описаны в блоке, сохраняют свой прежний смысл.

В момент выхода из блока (через конец или оператор перехода) все идентификаторы, которые описаны в этом блоке, теряют свой локальный смысл.

Следует описывать все идентификаторы, за исключением меток, идентификатора процедуры-кода (КОД) и идентификаторов стандартных функций. В начале любого блока никакой идентификатор, за исключением случаев, указанных в разд. 1.2.5.3, *б,* нельзя описывать более одного раза.

Синтаксис

<описание> : : =<описание типа>|<описание массива> | <описание составной переменной>|<описание составного массива> | <описание переключателя>

1.5.1. Описания типа

1.5.1.1. Синтаксис

<постоянный повторитель > : : = (<целое без знака >)
<переменный повторитель> : : = (<>идентификатор>)
<9 позиция> : : =9|9<постоянный повторитель>
<1 позиция> : : =l|1 <постоянный повторитель>
<повторитель>:: = <постоянный повторитель> | <переменный повторитель>
<С позиция> : : =С|С<повторитель>
<формат целого без знака>::=<9 позиция> | <формат целого без знака>|<9 позиция >
<формат целого> : : =<формат целого без знака>|+<формат целого без знака>| — <формат целого без знака>
<формат порядка> :: $=_{10}$+9
<формат правильной дроби> : : = ·<формат целого без знака>
<формат дроби>: : = <формат правильной дроби>|<формат целого без знака><формат правильной дроби>
<формат вещественного без знака> : : =<формат дроби>|<формат дроби><формат порядка>
<формат вещественного> : : =<формат вещественного без знака>|+<формат вещественного без знака>| — <формат вещественного без знака>

<указание вида числа> : : =<формат целого> | <формат вещественного>| < 1 позиция >
<указание вида строки> :: =<С позиция>
<указание вида> : : =<указание вида числа> | <указание вида строки>
<элемент списка типа> : : =<идентификатор>| <идентификатор> **вид**<указание вида>
<список типа> : : =<элемент списка типа> | <список типа>,<элемент списка типа>
<тип> : : = **целый** | **вещественный** | **строчный**
<описание типа> : : =<тип> <список типа>

1.5.1.2. Примеры

целый P, H, B **вид** 9 (5)
строчный ДАТА **вид** C (17)
вещественный A **вид** 9.9999, B1, B2, P **вид** 9**.** 9_{10}+9
строчный ДЕТАЛЬ **вид C (N),** ФАМИЛИЯ **вид C** (H)

1.5.1.3. Семантика

Описание типа служит для указания того, что некоторые идентификаторы представляют простые переменные данного типа. Переменные, которым описанием дан тип **вещественный,** могут принимать только положительные и отрицательные действительные значения, включая нуль. Переменные, которым описанием дан тип **целый,** принимают положительные и отрицательные целые значения, включая нуль. Переменные, которым предписан тип **строчный,** принимают значения, являющиеся открытыми строками.

1.5.1.3.1. Указание вида. Все переменные типа **строчный** должны иметь в описании указание вида строки. Переменные типа **целый** и **вещественный** могут иметь в описании указание вида числа.

Переменные, имеющие в описании указание вида, трактуются как величины, значения которых имеют фиксированную длину и фиксированный состав символов в заданных позициях. Указание вида характеризует только ту переменную, за идентификатором которой оно непосредственно следует.

1.5.1.3.2. Указатели символов. Указатель «9» означает, что в данной позиции значения числовой величины может стоять одна из десятичных цифр: 0, 1, 2, 3, 4, 5, 6, 7, -8, 9.

Указатель «1» предписывает, что в данной позиции числа может стоять двоичная цифра: либо 0, либо 1.

Указатель «.» предписывает помещение в данном разряде десятичной точки (в явном виде).

Указатель «$_{10}$» предписывает помещение в данном разряде символа $_{10}$.

Указатель « + », использованный в начале указания вида числа, предписывает помещение в данном разряде, знака «плюс» для положительных чисел и знака «минус» — для отрицательных чисел. Аналогичное действие вызывается в том случае, когда он стоит после указателя «$_{10}$».

Указатель «—», использованный в начале указания вида числа, предписывает помещение в данном разряде знака «пробел» (_) для положительных чисел и знака «минус» — для отрицательных чисел.

Указатель «С» допускает любой строчный символ в данной позиции значения строчной переменной.

1.5.1.3.3. Повторитель показывает, сколько раз должен быть повторен символ, стоящий непосредственно, перед ним. Так, указания видов +9999 и +9 (4) эквивалентны. Число символов в указании вида (повторитель содержит три символа) не должно превышать девяти. В переменном повторителе допускается использование простой переменной типа целый, не имеющей указания вида. Указанная переменная не должна быть локальной в блоке, для которого имеет силу соответствующее описание.

Указание вида	Запись числа
+99.999	—12.387
99.99	12.39
999.999	012.3870
—999.99	—012.39

1.5.1.3.4. Общее число указателей символов (с учетом повторителей) определяет длину слова. Эта длина для переменных типа **целый** и **вещественный** не должна превышать девяти символов (если указание вида задается с помощью указателя символа 9) или тридцати шести символов (если указание вида задается с помощью указателя символа 1).

1.5.1.3.5. Примеры действия указаний вида. В качестве примера используется число — 12.387 (см. табл. выше)

1.5.2. Описания массивов

1.5.2.1. Синтаксис

<нижняя граница> :: = <целое без знака>1 — <целое без знака> | <идентификатор>
<верхняя граница> : : =<целое без знака>| — <целое без знака> | <идентификатор>
<граничная пара> : : =<нижняя граница> : <верхняя граница>
<список граничных пар> : : =<граничная пара> | <список граничных пар>, <граничная пара>
<идентификатор внешнего массива> : : =Я <адрес>
<элемент описания массивов> : : = <идентификатор массива>**вид**<указание вида>|<идентификатор массива> | < идентификатор внешнего массива> **вид<**указание вида> | идентификатор внешнего массива >
<сегмент массивов > : : =<элемент описания массивов> [<список граничных пар>] | <элемент описания массивов>,<сегмент массивов>
<список массивов>: : = <сегмент массивов >|<список массивов>,<сегмент массивов >
<описание массивов> : : =**массив**<список массивов>| **целый массив** <список массивов>| **строчный массив**<список массивов>

1.5.2.2. Примеры

массив А, В, Я00036000 [7:М, 2:Н], Р[—2:10]
целый массив А вид 9 (7) [1 : 20]
строчный массив В1 **вид** С (15) [—2: Р], А **вид** С, К5
вид С (2), Н **вид** С (7) [1:К]

1.5.2.3. Семантика

В описании массивов определяется, что один или несколько идентификаторов (или идентификаторов внешних массивов) представляют многомерные массивы переменных с индексами, и задаются размерности массивов, границы индексов, типы и, возможно, виды переменных. Если в описании массива есть указание вида, то задаются длина и состав символов значений переменных с индексами (см. разд. 1.5.1.3.2). Указание вида распространяется на идентификатор массива, стоящий непосредственно перед таким указанием.

1.5.2.3.1. Границы индексов. Границы индексов массива даны в первых индексных скобках, следующих за идентификатором
данного массива (или идентификатором внешнего массива), в виде списка граничных пар. Каждая граничная пара этого списка дает нижнюю и верхнюю границы индекса. Список граничных пар дает границы всех индексов в порядке их перечисления слева направо.

1.5.2.3.2. Размерности. Размерности определяются как число граничных пар в описке граничных пар.

1.5.2.3.3 Типы. Все массивы, данные в одном описании, имеют один и тот же предписанный им тип. Если тип отсутствует, то подразумевается тип **вещественный.**

1.5.2.4. Переменные, использованные в качестве границ индексов

Простые переменные, использованные в списке граничных пар, должны по описанию иметь тип **целый**, а также не содержать в описании указания вида. Кроме того, они не должны быть локальными в том блоке, для которого имеет силу описание массивов. Из этого следует, что в самом внешнем блоке программы могут быть описания массивов только с постоянными границами.

1.5.2.5. Идентичность переменных с индексами

Значения переменных с индексами будут в любой момент времени определены только для той части этих переменных, у которых значения индексов лежат в пределах границ индексов, вычисленных в последний раз.

1.5.2.6. Внешние массивы

Идентификатор внешнего массива может быть использован только в качестве фактического параметра процедуры-кода со спецификациями 'ЗАПИСЬ' и 'ЧТЕНИЕ' (см. разд. 2.17.9), причем < адрес> представляет собой истинный начальный адрес внешнего массива на магнитной ленте.

1.5.3. Описания составных переменных и составных массивов

1.5.3.1. Синтаксис

<элемент описания> : : =<описание типа> | <описание массивов> | <описание составной величины>
<составная величина> : : = <идентификатор составной переменной>| **массив <**идентификатор составного массива>[<список граничных пар>]
<структура составной величины> : : = <элемент описания> **уровень** |<элемент описания>; <структура составной величины>
<описание составной величины> : : **=составной** <составная величина>;<структура составной величины>
<идентификатор внешней составной> : : =Я < адрес >
<идентификатор внешнего составного массива> : : =Я <адрес>
<внешняя составная величина> :: =<идентификатор внешней составной>| **массив**<идентификатор внешнего составного массива> [<список граничных пар>]
<указатель начала> : : =<целое выражение>
<указатель длины> : : =<целое выражение>
<описание внешней составной> : : **=составной**<внешняя составная величина>;<указатель начала>, <указатель длины**>уровень**

1.5.3.2. Примеры

составной А; строчный В**О вид** С (15);
целый В**;**
составной В1**; целый** Д **вид** 99**,** Д1 **вид** 99
уровень;
массив Н **вид** 9.9 [1:5]
уровень
составной массив ВЕДОМОСТЬ [1 :300];
целый ТАБЕЛЬНЫЙ НОМЕР **вид** 9 (б);
составной ДАТА;
целый ДН, МЦ, ГД **уровень;**
целый НОМЕР СТАНКА **вид** 999**;**
вещественный ОБЩАЯ СУММА
уровень
составной массив Я03007000 [1 :500]; В2, А1 **уровень**
составной Я01150500**;** 250, 1000 **уровень**
составной массив Я01004000 [1 : М]; Р+Н, 600 **уровень**

1.5.3.3. Семантика

Как следует из синтаксиса, описание составной величины может быть двух видов:

составной <идентификатор> ;<структура составной величины>
или
составной массив<идентификатор>[<список граничных пар>] ;<структура составной величины >

Первое определяет составную переменную, а второе — составной массив.

В описании составной переменной перечисляются идентификаторы, первый из которых. именует составную переменную, а остальные — переменные или массивы (элементарные или составные), входящие в ее состав. Кроме того, задается последовательность расположения величин в составной переменной и уровни их иерархии.

Составной массив — это массив, состоящий из одинаковых по структуре составных переменных.

В указателе начала и указателе длины внешней составной допускается использование целых выражений. Переменные, входящие в состав этих выражений, не должны быть локальными в блоке, для которого имеет силу описание внешней составной.

Указатель начала представляет собой сдвиг относительно начального адреса, указанного после буквы Я.

Указатель длины представляет собой длину (в ячейках) одного элемента внешнего составного массива или длину внешней составной переменной.

1.5.3.3.1. Типы и виды. Составная переменная и составной массив не имеют типа и вида, так как в общем случае состоят из величин различных типов. Только элементарная величина может иметь тип и вид, что и указывается в описании на соответствующем уровне.

1.5.3.3.2. Поскольку элементами составных переменных и составных массивов, в свою очередь, могут служить массивы, то к ним относится все сказанное выше об описаниях массивов (см. разд. 1.5.2).

1.5.3.3.4. Обращение к элементам составных величин. Обращение к элементу составной переменной или составного массива начинается с элементарной переменной (см. разд. 1.1.1.1) и содержит в качестве уточнений (см. разд. 1.3.1.1) все промежуточные уровни иерархии, включая самый внешний. Например:

МЦ. ДАТА. ВЕДОМОСТЬ
или
НОМЕР СТАНКА. ВЕДОМОСТЬ [J+2]

1.5.3.3.5. Внешние составные величины. Идентификаторы внешних составных величин могут быть только фактическими параметрами оператора процедуры-кода со спецификациями 'ЗАПИСЬ' и 'ЧТЕНИЕ'.

1.5.4. Описания переключателей

1.5.4.1. Синтаксис

<переключательный список>:: = <метка> | <переключательный список>,<метка>
<описание переключателя>:: =переключатель<идентификатор переключателя> : =<переключательный список>

1.5.4.2. Примеры

переключатель П : =П1, П2, П3, П4
переключатель НI : =Р1, Т

1.5.4.3. Семантика

В описании переключателя задается совокупность значений соответствующих указателей переключателей. Эти значения даны одно за другим в виде меток, перечисленных в переключательном списке. С каждой из них связано положительное целое число 1,2. ..., получаемое при пересчете членов в описке слева направо. Значением указателя переключателя, соответствующим данному значению индексного выражения (см. разд. 1.3.8), является метка в переключательном списке, имеющая это данное значение индексного выражения своим порядковым номером.

1.5.4.4. Влияние области действия

Если указатель переключателя встречается вне области действия метки, входящей в состав переключательного списка, и если при вычислении значения этого указателя переключателя выбирается именно эта метка, то противоречие между идентификатором этой метки и идентификаторами, описания которых имеют силу там, где расположен указатель переключателя, устраняется путем соответствующих систематических изменений последних идентификаторов.

ГЛАВА 2
СОДЕРЖАТЕЛЬНОЕ ОПИСАНИЕ ЯЗЫКА АЛГЭМ

2.1. ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА

Основными элементами алгоритмического языка являются: символы, слова, выражения и операторы. Эти элементы представляют собой структуры нарастающей сложности, ибо слова: образуются из последовательно расположенных друг за другом символов, выражения состоят из групп слов, а операторы состоят из комбинаций выражений и отдельных слов. Поэтому описание языка автоматического программирования включает в себя:

1) Перечисление допустимых символов;
2) Правила образования слов;
3) Описание выражений, имеющих смысл в данном языке;
4) Рассмотрение всех типов операторов, из которых состоит язык.

Символы языка — это основные, неделимые знаки, с помощью которых записываются тексты на этом языке, т. е. программы. При описании языка автоматического программирования необходимо перечислить символы, которыми можно пользоваться, и принять соглашение о том, что нельзя использовать иные символы, кроме тех, которые указаны в списке. Существенное ограничение, которое накладывается на символы, используемые в языке, связано с процессами ввода программ в машину.

Слова языка — это структуры, образованные непосредственно из символов языка и являющиеся обычно минимальными единицами, которые имеют смысл сами по себе. Слова языка автоматического программирования—это, например, идентификаторы, переменные, метки или числа.

Выражения—это группы слов, соединенных между собой некоторыми символами. Выражениям обычно приписываются значения. Например, арифметическое выражение $A + B$ имеет значение, равное сумме значений (величин) составляющих его переменных A и B. Если в какой-то момент времени переменные A и B имеют значения, равные, соответственно, 2 и —5, указанное арифметическое выражение в этот момент будет иметь значение, равное —3. Таким образом, выражение характеризуется тем, что состоит из группы слов, имеющей общий смысл, а именно «значение».

Оператор языка автоматического программирования — это минимальная структура, представляющая законченное действие, т. е. задающая описание некоторого процесса (в этом смысле описания языка также являются операторами). Оператор аналогичен команде в машинном языке или предложению в естественном языке. Подобно тому, как для конкретной машины существует конечное число различных операций, которые могут быть выполнены с помощью команд, в языке автоматического программирования существует конечное число типов операторов.

Поскольку данная глава представляет собой руководство по программированию на АЛГЭМе, а не формальное описание этого языка (данное в гл. 1), описание перечисленных выше элементов языка будет дано в последовательности, определяемой методическими соображениями, а не соображениями, диктуемыми строгостью изложения.

2.2. ОСНОВНЫЕ СИМВОЛЫ

Основные символы языка АЛГЭМ следующие: буквы, цифры, знаки арифметических операций, операций отношения, операций следования, разделители, скобки, описатели.

2.2.1. Буквы

Заглавные русские и недостающие по написанию заглавные латинские буквы

АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЭЮЯDFG I J L N Q R S U V W Z .

Необходимо отметить, что в алфавите языка АЛГЭМ отсутствует буква ъ. Это связано с особенностями телеграфного аппарата СТА-2М, используемого для перфорации программ, записанных на АЛГЭМе.

2.2.2. Цифры

0 1 2 3 4 5 6 7 8 9

2.2.3. Знаки арифметических операций

+ - × / ↑

Последний символ обозначает операцию возведения в степень (см. также разд. 2.4.4).

2.2.4. Знаки операций отношения

$< \leq = \geq > \neq$

2.2.5. Знаки операций следования

на если то для цикл ко стоп

2.2.6. Разделители

, . $_{10}$: ; : = _ шаг до примечание

2.2.7. Скобки

( )
[ ]
‘ ‘

начало составной **конец уровень**

2.2.8. Описатели

целый вещественный вид массив переключатель строчный

Назначение перечисленных выше основных символов и травила их использования будут поясняться в ходе дальнейшего изложения.

2.3. ИДЕНТИФИКАТОРЫ

Идентификаторы — это наименования (имена), которые даются различным величинам. Идентификаторы представляются последовательностями букв и цифр, начинающимися с буквы. Например:

A ХУ1001
B12 КЛЮЧ

Примеры слов, не являющихся идентификаторами:

12ПР
0.5
A+B×(I + C)

Идентификаторы применяются для обозначения таких величин, используемых в программах, как переменные, метки, массивы и т. п. В АЛГЭМе различаются два класса идентификаторов.

2.3.1. Идентификаторы внутренних величин

Идентификаторы внутренних величин в дальнейшем будут называться просто идентификаторами. Эти идентификаторы используются для обозначения меток и тех величин программы, которые в процессе выполнения программы располагаются в оперативной памяти ЭВМ (т. е. для обозначения внутренних величин программы).

Идентификатор внутренней величины может быть любой длины, т. е. может состоять из любого количества букв и цифр. При этом идентификация (распознавание) обозначаемых этим идентификатором объектов производится по всем символам, составляющим указанный идентификатор.

2.3.2. Идентификаторы внешних величин

Идентификаторы внешних величин используются для обозначения тех величин программы, которые в процессе ее исполнения располагаются на магнитных лентах. Идентификаторы внешних величин отличаются от прочих идентификаторов тем, что начинаются с буквы Я, за которой всегда следуют восемь восьмеричных цифр.
Примеры:

Я01004000
Я03070000

2.4. АРИФМЕТИЧЕСКИЕ ВЫРАЖЕНИЯ

Арифметические выражения строятся из операндов (т. е. величин, участвующих в вычислениях), знаков арифметических операций и круглых скобок согласно обычным правилам. Каждое арифметическое выражение задает правило для вычисления одного числового значения. Если это числовое значение имеет тип **целый,** то определяющее его выражение есть целое выражение; если числовое значение имеет тип **вещественный,** то определяющее его выражение есть вещественное выражение.

Операндами целых и вещественных выражений являются числа, переменные и указатели стандартных функций.

2.4.1. Числа

Величина, значение которой в выражении задано ее написанием, называется константой. Во входном языке различаются числовые константы (или числа) и строчные константы (или строки — см. разд. 2.5).

Числа бывают двух типов: типа **целый** и типа **вещественный.** Числа типа **целый** могут быть операндами целых выражений, а числа типа **вещественный**—операндами вещественных выражений. . Тип числа определяется совокупностью цифр и прочих символов, из которых оно составлено. Любое число, записанное при помощи символов + — 0 1 2 3 4 5 6 7 8 9 есть число типа **целый.** Например:

-10 5
-0 12500

Любое другое число имеет тип **вещественный,** хотя его значение может быть целым, например, 3.0. Заметим, что существует различие между целым числом типа **вещественный** и числом типа **целый.** Число 3.0 может быть

представлено в ячейке числом, которое отлично от 3.0 вследствие погрешностей машинного представления чисел. Число 3 представляется точно.

Мантиссой называется последовательность цифр, перед которой стоит точка:

.256
.01

Десятичным числом называется целое число без знака, за которым следует мантисса:

2.56
0.2
356.10

Числа типа **вещественный** представляет собой либо мантиссу, либо десятичное число, либо десятичное число, за которым следует символ $_{10}$ и одна из десятичных цифр со знаком или без знака, \ либо символ $_{10}$, за которым следует десятичная цифра со знаком или без знака. Число типа **вещественный** может иметь перед собой знак «плюс» или «минус».

Десятичное число в языке АЛГЭМ обязательно должно иметь мантиссу даже в том случае, если ее значение равно 0,

например, 13.0

Точка может стоять в начале числа, но не может стоять в конце.

Очень большие и очень малые числа в обычной математической записи изображаются при помощи масштабного множителя, например, $3.4_{10}9$ обозначает $3.4\times10\uparrow9$, где знак ↑ означает возведение в степень.

Примеры чисел типа **вещественный:**

0.5103
—10—7
+ 127.010+8

Примеры неправильной записи чисел:

125_{10}—3 (запрещается использовать целое число перед символом$_{10}$);
1.7_{10}l9 (в качестве показателя степени масштабного множителя может быть только одна цифра);
1.135_{10}+ ·3 (в качестве показателя степени масштабного множителя можно использовать только цифру).

Диапазоны представления целых и вещественных чисел в АЛГЭМе соответствуют диапазонам представления соответствующих чисел в ячейке ЭВМ «Минск-22».

2.4.2. Переменные

Величина, значение которой в выражении задано ее наименованием (идентификатором), называется *переменной.* Если наименованием переменной служит один идентификатор, то такая переменная в языке АЛГЭМ называется простой переменной. Например:

A17
БАЛАНС
ABC

Большинство программ имеет дело с последовательностями величин, которые обрабатываются одинаково. Было бы очень неудобно применять индивидуальные наименования для всех величин, образующих последовательность. Поэтому такие величины объединяются в массив. Всему массиву присваивается одно наименование, а отдельные его элементы различаются при помощи индексов. Так как индексы нельзя писать в виде подстрочных символов, для них вводятся специальные обозначения: индексы заключаются в квадратные скобки. Идентификатор массива, за которым следует описок индексных выражений, заключенный в квадратные скобки,— это переменная с индексами. В качестве индексного выражения может употребляться любое арифметическое выражение, в частности и такое, которое само содержит переменную с индексами. Ограничений на глубину индексации не накладывается. Индексные выражения разделяются запятыми. Их количество определяется размерностью массива, элементом которого является данная переменная с индексами (см. раздел 2.16).

Для нахождения элемента массива, который указывается с помощью переменной с индексами, вычисляются фактические числовые значения арифметических выражений, определяющих индексы переменных. Эти числовые значения должны иметь тип **целый**. Поэтому в том случае, когда индексным выражением служит вещественное выражение А, его числовое значение округляется до ближайшего целого числа по формуле:

entier (A+ 0.5)*

Примеры переменных с индексами:

AB [5, 1.3]
L[sin (H↑ P1/2.0), T3, 5, I1+ E]
B1 [A[ТАБЕЛЬ[I]]]

Порядок расположения переменных с индексами внутри массивов соответствует лексикографическому порядку значений индексов** т. е.

A [K_1, K_2, ..., K_n] предшествует A [M_1,M_2, . . , M_n],

если $\left\{\begin{array}{c} K_i = M_i (i = 1,2,...,P-1) \\ K_p < M_p \end{array}\right\}$ (1≤P≤n)

---

* См. раздел 2.4.3. Указатели стандартных функций.

** Этим правилом следует руководствоваться также при подготовке массивов исходной информации.

Тип значения простой переменной определяется описанием этой переменной; тип значения переменной с индексами определяется описанием соответствующего идентификатора массива (см. раздел 2.16). Значение переменной с индексами определено только в том случае, когда значения индексных выражений находятся в пределах границ индексов массива (см. раздел 2.16.2). Например, если описание массива имеет вид

массив A [1:100,2:20],

то значения переменных с индексами

A [-15,2.9] и A [215, 17]

не определены.

Простые переменные и переменные с индексами называются элементарными переменными. Элементарные переменные могут входить в выражения со ссылкой на более крупные величины, в состав которых они входят (согласно описанию). Такие ссылки называются уточнениями (см. разд. 2.16), и используются для исключения двусмысленностей, связанных с возможным использованием одного и того же идентификатора для обозначения переменных, входящих в разные составные величины. Уточнения отделяются друг от друга и от элементарной переменной, которую они поясняют, точками. Выписывать следует все уточнения в порядке их иерархии.

Пример:

A3. ТАБЕЛЬ [1,2]
ГРАФА [1]. ТАБЛИЦА [N+2, S]. ВЕДОМОСТЬ

Тип значения элементарной переменной, записанной с уточнениями, определяется описанием этой переменной, которое находится в описании более крупной величины, в состав которой входит элементарная переменная, на соответствующем уровне (см. раздел 2.16).

Переменные типа **целый** могут быть операндами только целых выражений; переменные типа **вещественный** могут быть операндами только вещественных выражений. Исключение составляет случай, когда переменные являются аргументами стандартных функций или образуют индексные выражения.

2.4.3. Указатели стандартных функций

В АЛГЭМе введены следующие стандартные функции:

abs	для обозначения абсолютной величины
знак	для обозначения знака величины
sgrt	для обозначения квадратного корня (положительного значения)
sin **cos** }	аргументы задаются в радианах
arctg	вычисляются значения на отрезке $[-\pi/2, +\pi/2]$
ln	для обозначения натурального логарифма
exp	для обозначения показательной функции по натуральному основанию

Для удобства трансляции (с тем, чтобы идентификатор функции воспринимался как единый символ) идентификаторы стандартных функций в АЛГЭМе необходимо подчеркивать. Аргументы стандартных функций заключаются в круглые скобки. Нельзя писать **sin** X, следует писать **sin** (X).

Смысл указателей стандартных функций в большинстве случаев очевиден. Стандартная функция знак определяется следующим образом:

$$\textbf{знак}\ (A) = \begin{cases} +1, \text{если } A>0 \\ 0, \text{если } A=0 \\ -1, \text{если } A<0 \end{cases}$$

Идентификатор стандартной функции, за которым следует заключенное в круглые скобки арифметическое выражение, например, sin (X), называют указателем стандартной функции. Указатели стандартных функций определяют отдельные числовые значения. Указатели всех стандартных функций, кроме **abs** (A) и **знак** (A), определяют числовые значения типа **вещественный** и могут быть операндами только вещественных выражений. Они определены в том случае, если в качестве аргумента используются вещественные выражения.

Функции **abs** (A) и **знак** (A) определены для A, являющегося как целым, так и вещественным выражением. При этом тип значения, определяемого указателем стандартной функции **abs** (A), совпадает с типом значения выражения A.

Указатель стандартной функции знак (A) независимо от типа выражения, являющегося аргументом, всегда определяет значение типа целый.

Для преобразования значения типа **вещественный** в значение типа **целый** введена стандартная функция преобразования **entier** (A). Указатель функции **entier** (A) определяет значение типа целый, являющееся наибольшим целым числом, не превосходящим значения вещественного выражения A.

Примеры указателей функций:

ln (E+A+B)
entier (**sin**(B))
abs (A-B)
entier (2.6)*
entier (—2.6)**

* Значение функции равно 2

** Значение функции равно —3

2.4.4. Знаки операций и скобки

Для обозначения операций употребляются знаки арифметических операций:
+ сложение
— вычитание
× умножение
/ деление
↑ возведение в степень.

Круглые скобки применяются обычным образом. Квадратные скобки можно использовать только для указания индексов. Так выражение

$$[(A+B)\times(C + D)]\times(X+У)$$

недопустимо. Его следует записать в виде:

$$((A+B)\times(C+D))\times(X+У).$$

Следует иметь в виду следующее:

знак × в выражении не заменяется точкой и не опускается;

два знака арифметических операций никогда не пишутся рядом (например, при переносе на следующую строку);

знак / в целом выражении не допускается.

2.4.5. Очередность выполнения операций в арифметических выражениях

1) Прежде всего применяется правило скобок.
2) Очередность выполнения операций внутри одних круглых скобок устанавливается следующим образом:
   ↑
   × /
   + —
3) Если операции имеют одинаковый порядок старшинства, например, A × B / C, то операция, расположенная слева, выполняется раньше.

Примеры:

A+B-C	означает	(A+B)-C
A+B↑C+D	»	(A+(B↑C)+D)
A/B×C	»	(A/B)×C, но не A/(B×C)
—A↑B	»	—(A↑B), но не (—A)↑B.

2.4.6. Знак операции ↑ в целом и вещественном выражениях

В целом выражении основанием степени может служить:
число типа **целый** без знака;
переменная типа **целый;**
указатели стандартных функций:

знак (A**),**
abs (A**), где** A—целое выражение,
entier (A**);**

заключенное в круглые скобки целое выражение.

Показателем степени может служить целое число без знака. Согласно синтаксису языка в целом выражении повторное возведение в степень не допускается.

Примеры ошибочных записей:

abs (E/F)↑ 13 можно записать в виде:
entier (abs (E/F))↑13
(1+2×P—K)↑3↑|17 можно записать в виде:
((1+2×P—K)↑3)↑17

В вещественном выражении в качестве основания степени может употребляться:
число типа **вещественный** без знака;
переменная типа **вещественный;**
указатели стандартных функций:

sgrt (A**),**
sin (A),
cos (A),
arctg (A),
exp (A),
ln (A),
abs (A), где A—вещественное выражение;

заключенное в круглые скобки вещественное выражение.

В качестве показателя степени в вещественном выражении допускается любая из величин, перечисленных в п. п. 1—4.

Значение выражения A↑B равно **exp (**B×**ln**(A)) в случае если A>0, и не определено, если A≤O.

Рекомендуется писать:

A×A вместо A↑ 2.0
1/(A×A×A) вместо A↑ (—3.0)
и т. д.

Примеры:

Р.Н.А [I, J]↑ cos (У+Z×3,0)
Е↑W [1+2,8] ↑ (А+3.0)

2.4.7. Знак операции / в вещественном выражении

Вещественным множителем называется либо каждая из перечисленных в предыдущем разделе конструкций (1—4), либо комбинация вида А↑В, где А — вещественный множитель, В — показатель степени.

В качестве делителя в вещественном выражении должен стоять вещественный множитель, значение которого не равно нулю. Если значение вещественного множителя равно нулю, результат операции не определен.

Примеры целых выражений:

entier (1.5 + A/B)+A. РЕГ[I+0.5]
abs (РЕЗУЛЬТАТ) × **знак** (РЕЗУЛЬТАТ)
(215-(А×В)↑5)↑2 + П [1 + 1.7, J]

Примеры вещественных выражений:

(2.2+**cos** (А) **) / sgrt** (А↑2.0+1.0)
exp (2.5)—$_{10}$—7
2.0×А+Х×У+0.5

При использовании вещественных выражений необходимо иметь в виду следующее: машинное представление вещественного числа является приближенным как вследствие конечной длины ячейки, так и вследствие погрешностей программы перевода из десятичной системы в двоичную. Поэтому при вычислении значений вещественных выражений может возникнуть отклонение от математически определенного результата. Контроль точности вычислений, если это необходимо, должен производиться методами численного анализа и рассматриваться как часть описываемого вычислительного процесса. (Примеры см. в разд. 2.10. и 2.12.)

2.5. СТРОЧНЫЕ ВЫРАЖЕНИЯ

Строчное выражение ,в АЛГЭМе—это либо строка, либо переменная типа **строчный**. Строка также имеет тип **строчный.** Элементами строки могут быть строчные символы и строки. Строка состоит из цепочки элементов строки, заключенной в строчные кавычки. Строчными являются следующие основные символы:

буквы, цифры, знаки арифметических операций, знаки операций отношения, символы $_{10}$ |

Символ _ означает пробел и не имеет смысла вне строк. Символ | используется как вертикальный разделитель при оформлении документов для выдачи на АЦПУ (печатается как « : »).

Примеры строчных выражений:

К17 —переменная типа **строчный**
'17_АПРЕЛЯ_ 'АРГ+12.05'_ 1970_ Г.'-строка
"ПЕТРОВ _ _'СТАЖ""-строка

Значением строчного выражения является открытая строка. Открытая строка — это цепочка элементов строки, освобожденная от внешних строчных кавычек.

Значениями строчных выражений для приведенных выше примеров (второй и четвертой) являются открытые строки:

17_ АПРЕЛЯ_ 'АРГ+12.5'_ 1970_ Г.

'ПЕТРОВ _ _'СТАЖ"

2.6. ОПЕРАТОРЫ ПРИСВАИВАНИЯ

Оператор присваивания — это основной оператор языка, который присваивает значения одной или нескольким переменным. Оператор присваивания имеет вид в простейшем случае:

<переменная> : =<выражение>

Такая запись означает следующее: заменить значение переменной, стоящей слева, значением выражения, стоящего справа. Комбинацию символов := следует рассматривать как один символ; его называют знаком присваивания. Не следует смешивать его со знаком равенства.

Примеры:

ТАБЕЛЬ. А [I] : =Е×В [I, J] +**ln** (Z)
К: =К-5
ФАМИЛИЯ. СПИСОК [К] :='ИВАНОВ'

Второй пример показывает, что символ : = означает «заменить на», а не «равно».

Один оператор присваивания может присвоить одно и то же значение нескольким величинам. Такой многократный оператор присваивания можно проиллюстрировать примерами:

А: =Е [I,M + I] : =В.С: =0
Р.Н : =М : =ОТЧЕТ : '_ _ _ - _ _ _ '

Все переменные, стоящие слева от выражения, должны быть одного и того же типа.

Допускаются все возможные комбинации типов переменных списка левой части оператора присваивания и выражения правой части:

тип переменных списка левой части:	*тип выражения:*
целый	**целый**
целый	**вещественный**
целый	**строчный**
вещественный	**вещественный**

вещественный	**целый**
вещественный	**строчный**
строчный	**строчный**
строчный	**целый**
строчный	**вещественный**

Выполнение оператора присваивания происходит в общем случае в три этапа:

1) значения всех индексных выражений, встречающихся в переменных левой части, вычисляются в порядке слева направо;

2) вычисляется значение выражения;

3) значение выражения присваивается всем переменным списка левой части, при этом индексные выражения имеют значения, вычисленные на первом этапе.

Если необходимо, происходит изменение типа значения выражения в соответствии с типом переменных описка левой части. Такой порядок выполнения оператора присваивания необходим для того, чтобы исключить неоднозначное толкование результата в тех случаях, когда выражение, стоящее в правой части оператора присваивания, содержит переменную из списка левой части или переменную, входящую в индексное выражение переменной списка левой части. Например, многократный оператор присваивания

I: =A [I] : = I-2

можно записать с помощью простых операторов присваивания (согласно пунктам 1—3) таким образом:

L:=I; I: =I—2; A[L]:=I-2;

где L — переменная типа целый.

2.6.1. Указание вида

С помощью оператора присваивания может осуществляться не только присваивание переменным списка левой части значения выражения, но и редактирование этого значения в соответствии с заданным форматом. Формат задается программистом в описании соответствующего идентификатора (в описании типа или описании массивов — см. раздел 2.16) и называется указанием вида. Переменные, имеющие в описании указание вида, трактуются как величины, значения которых имеют фиксированную длину и фиксированный состав символов в заданных позициях.

Состав символов задается следующими указателями символов:

9 + — . $_{10}$ С 1

Указатели символов 9, С и 1 могут быть многократно повторены. Количество повторений задается как в явном виде (для символа 9, например, 9999), так и с помощью так называемого повторителя — в круглых скобках задается количество повторений указателя символа. Например: 9(4), С(15), 1(34). Для указателя символа 9 возможно использование обеих конструкций: 9999 эквивалентно 9 (4). Для указателей символов С и 1 допускается использование только второго способа: С (15) или 1 (25). Для указателей символов 9 и 1 в качестве повторителя допускается использовать только целые числа без знака. Для указателя символа С можно использовать в качестве повторителя простую переменную типа **целый,** не имеющую согласно описанию указания вида.

Многократное повторение остальных указателей символов (+ — . $_{10}$) в указании вида не допускается.

Длина слова определяется общим числом указателей символов (с учетом повторителей), содержащихся в указании вида.

2.6.2. Указатели символов

Указатель «9» означает, что в данной позиции значения числовой величины может стоять одна из десятичных цифр: 0> 1, 2, 3, 4, 5, 6, 7, 8, 9.

Указатель « + », использованный в начале указания вида, предписывает помещение в данной позиции знака « + » для положительных чисел и знака «—» — для отрицательных чисел.

Указатель «—», использованный в начале указания вида числа, предписывает помещение в данной позиции знака «_» (пробел) для положительных чисел и знака «—»—для отрицательных чисел.

В состав указания вида переменной типа **целый** могут входить любые из символов „9" „ + " „—", причем „9" может использоваться с повторителем. Например:

999 +9 (5).9 -9999

В состав указания вида переменных типа **вещественный** могут входить любые из описанных выше указателей символов: „9" „ + " „— ” „ . ”„$_{10}$”.

Указатель «•» предписывает помещение в данной позиции числового значения десятичной точки (в явном виде).

Указатель «$_{10}$» предписывает помещение в данной позиции символа «$_{10}$».

Указатель « + », использованный после указателя «$_{10}$», оказывает такое же действие, как при использовании в начале указания вида.

Все, что было сказано в разделе 2.4.1 о строении чисел типа **вещественный,** относится и к строению указания вида для переменных типа **вещественный,** за исключением того, что не допускается конструкция $_{10}$ + 9 и после символа $_{10}$ всегда должен стоять +. Например:

99_{10}—9 неверная запись

99.9_{10}+9 верная запись

После символа „$_{10}$” следует указывать только один указатель символа „9" со знаком „ + "> что соответствует одной десятичной цифре в показателе масштабного множителя. Например:

$+999.9_{10}99$	неверная запись
$+999.9_{10}+9$	верная запись

Некоторые замечания, относящиеся к указанию вида для переменных типа **целый** и **вещественный:**

1) Каждому из перечисленных выше указателей символов в машинном представлении соответствует 4 двоичных разряда;

2) Длина слова, задаваемая указанием вида, не должна превышать девяти символов.

Пример:

+9(3) ·9 (6)

Такое указание вида недопустимо, т. к. общее число позиций, определяемое этим указанием вида, равно 10.

3) Отсутствие знака перед указанием вида означает, что и для положительных, и для отрицательных чисел (и показателей степени в масштабных множителях) после редактирования их в соответствии с заданным форматом знак должен быть опущен.

Указатель символа „1" предписывает, что в данной позиции числового значения типа **целый** или **вещественный** может стоять двоичная цифра: 0 либо 1. Длина слова, задаваемая с помощью указателя символа „1", не должна превышать тридцати шести символов. Каждому символу в ячейке оперативной памяти соответствует один двоичный разряд позиции значения. Указание вида, содержащее указатель символа „1", не должно содержать никаких других указателей символов.

Указатель символа «С» допускает любой символ (кроме круглых скобок) в данной позиции значения строчной переменной, если это значение строчной переменной вводится в оперативную память с перфоленты или с перфокарт. (Подробнее см. в разд. 2.17).

Указатель символа «С» допускает также любой строчный символ в данной позиции значения переменной типа **строчный** в случаях, если значение этой переменной определяется оператором присваивания со строкой в правой части.

Каждому указателю символа «С» в машинном представлении соответствуют 7 двоичных разрядов. Длина слова, задаваемая с помощью указателя символа «С», ограничена размером оперативной памяти, выделенной для переменных программы (7400_8 ячеек МОЗУ-2).

Число символов (с учетом круглых скобок и символов повторителя), с помощью которых записывается указание вида, не должно превышать девяти. В этом смысле пример +9(3) .9 (6) также ошибочен. Отметим, что символы повторителя принимаются во внимание только в том случае, если этим повторителем служит целое число без знака. Если повторителем является идентификатор переменной, его длина не принимается во внимание. Например:

С (УГОЛ ЗРЕНИЯ)

Здесь принимаются во внимание только три символа (С и две скобки). Тогда как в случае

С (27896547)

длина указания вида считается равной 11 символам.

2.6.3. Редактирование целых значений

Редактирование значений типа **целый** в соответствии с указанием вида осуществляется следующим образам: последняя (правая) позиция указания вида совмещается с последней (младшей) цифрой целого числа. Отредактированное число должно состоять из такого количества цифр, которое равно количеству позиций указания вида. Если число состоит из меньшего количества цифр, то вставляются старшие незначащие нули. Если число содержит больше цифр, чем позиций в указании вида, то левые (старшие) цифры отсекаются. Например:

указание вида	999	+9999	+9(5)
число 1375	375	+1375	+01375

Смысл редактирования по указателю символа «1» тот же, что по указателю 9, только в качестве цифр редактируются двоичные разряды, которыми представляется число в двоичной системе. Например: машинные слова 01110101 и 1001, отредактированные **в** соответствии с указанием вида 1(6), имеют вид: 110101 и 001001 (редактирование по знаку см. разд. 2.6.1**).**

2.6.4. Редактирование вещественных значений

1) В том случае, если в указании вида отсутствует указатель порядка $_{10}$, редактирование производится так: совмещается десятичная точка указания вида с десятичной точкой числа. Редактирование целой части вещественного числа осуществляется в соответствии с редактированием целых чисел. При редактировании дробной части делается округление: к разряду числа, соответствующему последней цифровой позиции указания вида, прибавляется 1, если цифра следующего за ним разряда не меньше 6. В противном случае округление не производится. Если дробная часть вещественного числа содержит меньше цифр, чем имеется в указании вида, то число дополняется справа нулями. Если редактируемое число имеет порядок, то десятичная точка перемещается в соответствии со знакам и величиной показателя степени порядка, т. е. вещественное число с порядком переписывается как десятичное число (вещественное число без порядка), а затем производится редактирование. Например:

указание вида	—99.99		
редактируемые числа	$.12_{10}+2$	+195.127	—127.0
они же после редакции	_12.00	_95.13	—27.00

2) Если в указание вида вещественного значения входит указатель символа „$_{10}$", то совмещается первая (левая) значащая цифра числа с первой (левой) цифровой позицией указания вида. Положение десятичной точки и

указателя символа „$_{10}$” в отредактированном числе определяется указанием вида, а величина и знак показателя степени масштабного множителя определяется, исходя из величины редактируемого числа. Округление производится, как указано выше, например:

указание вида	—9.9$_{10}$+9		
редактируемые числа	18.1$_{10}$—2	12.95	—5.98
они же после редакции	_1.8$_{10}$—1	_1.3$_{10}$+1	—6.0$_{10}$+0

При выдаче на ТБПМ вещественных чисел, отредактированных в соответствии с. указаниями видов, десятичная точка и символ „$_{10}$” заменяются пробелами. В тех случаях, когда в указании вида стоит указатель символа «—», а редактируемое число положительно, вместо знака «пробел» на ТБПМ печатается 0. Например, числа последнего примера на ТБПМ напечатаются в следующем виде:

+01_8_—100	+ 01_3_ +100	+—6_0_+000*

2.6.5. Редактирование строчных значений

При редактировании строчных значений сохраняется принятый в трансляторе принцип редактирования: количество символов, пересылаемых в переменную описка левой части оператора присваивания, определяется указанием вида принимающей переменной. Присваивание выполняется символ за символом слева направо.

Пример.

```
начало
  строчный A вид C (15), B вид C (10), C вид C (5);
  B:='ТРАНСЛЯТОР'; C:='SSSSS'; A:=B;
  C:=B; КОД   ('ПЧ  АЦПУ', A, B, C)
конец
```

В результате работы этой программы на АЦПУ отпечатаются строки:

```
ТРАНСЛЯТОРSSSSS
ТРАНСЛЯТОР
ТРАНС
```

2.6.6. Изменение типа значения выражения с помощью оператора присваивания

Выше упоминалось, что все переменные списка левой части должны иметь один и тот же тип. Тип переменных списка левой части может как совпадать с типом выражения, стоящего в правой части оператора присваивания, так и отличаться от него. Если тип выражения отличается от типа переменных списка левой части, то считается, что в конце второго этапа выполнения оператора присваивания автоматически включается функция преобразования, соответствующая преобразованию значения выражения к типу переменных списка левой части.

Ниже приводится перечень всех возможных преобразований типа значения с помощью оператора присваивания:

целый	**вещественный**
вещественный	**целый**
целый	**строчный**
вещественный	**строчный**
строчный	**вещественный**
строчный	**целый**

Для преобразования значения выражения A из типа **вещественный** в тип **целый** функция преобразования выдает результат, эквивалентный

entler (A+0.5).

Преобразование из типа **целый** в тип **вещественный** эквивалентно преобразованию числа из формы записи целых чисел в форму записи вещественных чисел (с плавающей запятой).

Если в левой части оператора стоят переменные типа **целый** или **вещественный,** а в правой части — строчная переменная, то все переменные описка левой части должны согласно описанию иметь указания видов. В момент присваивания производится преобразование значения из типа строчный в тип целый или вещественный, которое состоит в перекодировке семиразрядных кодов АЦПУ, соответствующих позициям значения правой части, в четырехразрядные двоично-десятичные коды. Полученная последовательность четырехразрядных кодов представляет собой значение типа **целый** или **вещественный** в зависимости от типа переменных списка левой части оператора присваивания. Для того, чтобы такой оператор присваивания имел смысл, необходимо следить за тем, чтобы значениями строчных переменных, стоящих в правых частях таких операторов, служили открытые строки, являющиеся фактически десятичными числами.

Пример:

```
начало целый A вид 9 (3); строчный B вид C (3);
  B: ='489';    A:=B;    КОД ('ПЧ _ АЦПУ', B);
  КОД ('ПЧ_ 10', A)
конец
```

В результате работы этой программы на ТБПМ и на АЦПУ будет напечатано 489.

Во всех случаях, когда в процессе присваивания участвует переменная типа **строчный** или строка, осуществляется присваивание символа за символом слева направо, и количество пересылаемых символов определяется принимающей переменной.

---

* Знак +, напечатанный слева, не относится к числу (см. разд. 2.17.1).

Если список левой части оператора присваивания имеет тип **строчный,** а выражение является арифметическим, то:

такой оператор присваивания не может быть многократным;

в качестве арифметического выражения может выступать только переменная типа **целый** или **вещественный,** имеющая согласно описанию указания вида. (В этом случае указатель символа „|" в указании вида не допускается.)

Преобразование значения из типа **целый** или **вещественный** в тип **строчный** состоит в перекодировке четырехразрядных кодов, соответствующих позициям значения правой части, в семиразрядные коды АЦПУ.

Последнее преобразование удобно применять в тех случаях, когда необходимо напечатать на АЦПУ какие-либо числовые значения (таблицы, документы и т. п.), поскольку такие символы как „ •", „$_{10}$", „_" печатаются только на АЦПУ. В этом случае предварительно следует выполнить присваивание числового значения (уже обработанного согласно указанию вида) переменной типа **строчный,** описанной так, чтобы количество указателей символов „С" в ее указании вида было равно общему количеству всех указателей символов в описании соответствующей переменной типа **целый** или **вещественный.**

Пример:

```
начало массив A [1 : 100];  вещественный A1 вид 999.99;
    строчный  A2  вид  C (6);   целый  K;    КОД  ('ВВОДЛ _ 10—2', A);
    для К : = 1 шаг 1 до 100 цикл начало
    A 1 : = A [ K ];  A2 : =A1;
    КОД ('ПЧ_АЦПУ', A2) конец
конец
```

При работе с переменными типа **целый** или **вещественный,** имеющими согласно описанию указание вида, следует иметь в виду следующее: каждое появление такой переменной в программе внутри оператора присваивания (в левой или правой его части) вызывает обращение к подпрограмме перевода из десятичной в двоичную систему (для переменных, встречающихся в правой части) или к подпрограмме перевода из двоичной в десятичную систему и редактирования по виду (для переменных, встречающихся в левой части оператора присваивания). Аналогичные действия вызывает появление в программе переменной с индексами, являющейся элементом массива, имеющего в описании указание вида. В этом случае также удлиняется рабочая программа, так как расчет местонахождения такой переменной выполняется с точностью до разряда (если массив описан с указанием вида, переменные в нем упакованы в соответствии с этим указанием вида). Поэтому указание вида в описании рекомендуется давать в следующих случаях:

для выдачи значений переменных на печать с соответствующей редакцией;

для экономии памяти:

Пример:

```
начало массив A вид 99.9 [1 : 3000]; ... конец
```

Для размещения такого массива в памяти будет использовано 1334_{10} ячейки (по 9 двоично-десятичных символов в ячейке).

2.7. ЛОКАЛЬНЫЕ И ГЛОБАЛЬНЫЕ ИДЕНТИФИКАТОРЫ

2.7.1. Идентификаторы, не являющиеся метками

Идентификаторы, используемые в блоке В (см. разд. 2.14), и обозначающие переменные, массивы, составные переменные, составные массивы и переключатели, можно разделить на две группы:

а) идентификаторы, описанные в блоке В;

б) идентификаторы не описанные в блоке В.

Первые называются идентификаторами, локальными в блоке В, а вторые —глобальными для блока В. Например:

```
A:начало вещественный A1; ...
B: начало вещественный B1;  ... конец;  ... конец
```

В этой программе А1 —глобальный идентификатор для блока В; В1 —локальный идентификатор в блоке В; в то же время А1 — локальный идентификатор в блоке А.

Идентификатор, локальный в блоке, определен только внутри этого блока.

2.7.2. Идентификаторы, являющиеся метками

Для того чтобы один оператор мог ссылаться на другой, необходимо иметь возможность опознавать операторы. Для этой цели используются метки. Меткой может быть любой идентификатор, т. е. последовательность букв и цифр, начинающаяся с любой буквы, кроме Я (см. разд. 2.3). Метка пишется перед помечаемым оператором и отделяется от него символом «: ». Помещение метки с двоеточием перед оператором служит ее описанием в наименьшем содержащем эту метку блоке, в котором, следовательно, она является локальной.

Пример:

```
M:A:=B; ...
```

2.8. ОПЕРАТОРЫ ПЕРЕХОДА

Оператор перехода в общем случае состоит из символов **на**, за которым следует именующее выражение, т. е.

имеет вид:

на <именующее выражение>

Оператор перехода прерывает естественную последовательность действий, задаваемую порядком написания операторов, определяя своего преемника по значению именующего выражения. Следующим выполняемым оператором будет тот, который имеет это значение в качестве своей метки.

Именующим выражением в простейшем случае может служить метка.

Пример:

X:=0; M:X:=X+1; **на** M;

В этой последовательности из трех операторов первый оператор присваивания выполняется один раз, а второй оператор, помеченный меткой M, выполняется неограниченное число раз (практически до останова по переполнению).

В более сложном случае именующее выражение является указателем переключателя, и тогда оно представляет собой правило для определения метки оператора. Указатель переключателя имеет вид:

P [<индексное выражение>],

где P — идентификатор переключателя.

В качестве возможного значения указателя переключателя в каждый конкретный момент может выступать единственная метка из совокупности меток, определяемой описанием переключателя.

2.9. ОПИСАНИЯ ПЕРЕКЛЮЧАТЕЛЕЙ

Описание переключателя имеет вид:

переключатель < идентификатор переключателя>: = <переключательный список >

Пример:

переключатель P:=M1, M2, M3, ПН, B2

Как и всякое описание (см. разд. 2.16), описание переключателя располагается в начале блока и имеет силу только в этом блоке с исключением тех внутренних блоков, в которых имеются описания, использующие тот же идентификатор.

Переключательный список представляет собой последовательность меток, отделенных друг от друга запятыми. Как отмечалось выше, указатель переключателя имеет вид:

<идентификатор переключателя> [<индексное выражение>]

Указатель переключателя является именующим выражением, определяющим метку, номер которой в переключательном списке, считая слева направо, совпадает со значением индексного выражения. Если в списке не найдется позиции с соответствующим номером, значение указателя переключателя не определено. В этом случае не определен и результат действия оператора перехода.

Пример 1. Пусть имеется описание переключателя

переключатель S:=M, P, Q

Тогда оператор перехода **на** S [I] определен лишь для значений I, равных 1, 2 или 3. В этих случаях оператор осуществляет переход к меткам соответственно A, P или Q.

Пример 2. Пусть надо вычислить для разных K значение функции $У=f_K(Z)$, где K=1, 2, 3, 4

$f_1(Z) = 3{,}2Z+0{,}1Z^2$

$f_2(Z) = 3{,}7Z+0{,}8Z^{2{,}1}$

$f_3(Z) = 4{,}02Z + 1{,}2Z^{2{,}3}$

$f_4(Z) = 5{,}39Z+ 1{,}56Z^{2{,}5}$

Последовательность операторов, выполняющих соответствующие вычисления, будет следующая:

начало переключатель F: =F1, F2, F3, F4; **вещественный** У, Z; **целый** K;
КОД ('ВВОДЛ_10-2', K, Z); **на** F [K];
F1:У:=3.2×Z+0.1×Z↑2.0; **на** M;
F2:У:=3.7×Z+0.8×Z↑2.l; **на** M;
F3:У:=4.02×Z+l.2×Z↑2.3; **на** M;
F4:У:=5.3×Z+l.56×Z↑2.5; M:КОД ('ПЧ _ 2—10', У)
конец

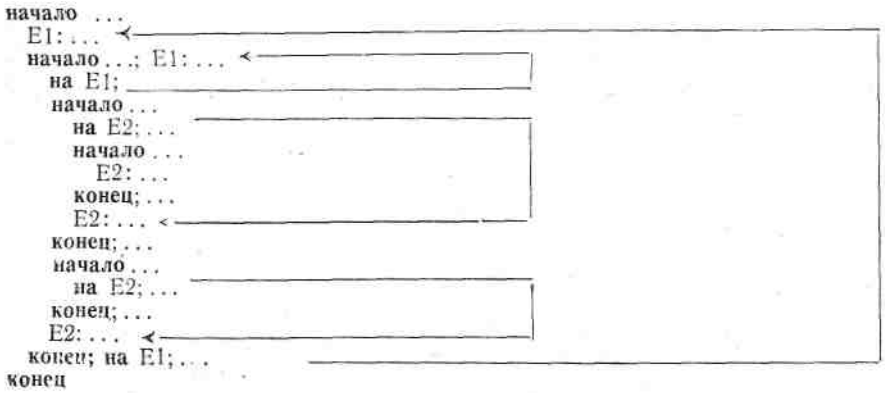
Пример 3. Дана последовательность операторов, снабженных метками L1, L2, ..., L12, M, L13. После операторов с метками L2, L7, L8 и L11 необходимо сделать обращение к одному и тому же оператору с меткой M. После каждого из таких обращений нужно возвращаться соответственно к операторам с метками L3, L8, L9 и L12.

Программа имеет вид:

начало переключатель П:=L3, L8, L9**,** L12, L13; ...
L1: ...; L2: ...; **I: = 1; на** M; L3: ...;
...; L7: **...**; **I:= 2; на** M; L8: ...;
I:= 3; **на** M; L9: .. .;
...; L11:.,.,.; **I:= 4; на** M; L12 : . . . ;
I: =5; M: **начало** . . . **конец; на** П [I]**;** L13:
конец

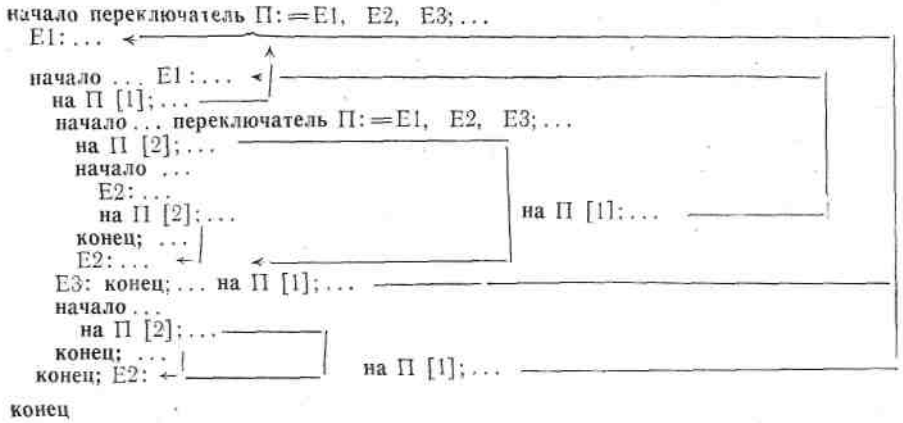
Выполнение оператора перехода в общем случае сводится к . отысканию идентификатора метки, служащей значением именующего выражения. После этого выполнение программы продолжается с оператора, помеченного этой меткой. Такой оператор отыскивается сначала в минимальном блоке, содержащем оператор перехода, затем (если такого оператора в минимальном блоке не окажется) в минимальном блоке, охватывающем этот блок, и т. д. При этом метки, находящиеся в блоках, вне которых расположен выполняемый оператор перехода, не принимаются во внимание. Эти метки не доступны для данного оператора.

Приведем пример, поясняющий некоторые возможные ситуации:



Во внутренних блоках некоторые идентификаторы меток, уже использованные во внешних блоках, могут использоваться повторно. Если из такого внутреннего блока указатель переключателя обращается к находящемуся во внешнем блоке переключательному списку, содержащему идентификаторы, изменившие свой смысл в этом блоке, то на время вычисления значения указателя переключателя восстанавливается прежний смысл этих идентификаторов. В приведенном ниже примере дается несколько возможных случаев использования указатели переключателя в операторах перехода.

Пример:



Указанные в этом примере переходы реализуются в следующей программе:

```
начало целый I;  переключатель П:=E1, E2, E3;
    I: = 1;  на M;   E3:I: = I+1;
    E1:КОД ('ПЧ_2-10', 111);
    если I = 3 то на M2;
    если I= 4 то на M3;
    M: начало целый J;  если I=1 то на П [3];
    E1:КОД ('ПЧ _ 2-10', 2); I:=I+1;
      начало переключатель П: =E1,  E2,  E3;
        на П [2];  M1:I: = I + 1;
        начало целый K; E2: КОД ('ПЧ_ 2-10', 3);
          если I = 4 то на П [1];
          на П [2]
        конец; E2:КОД ('ПЧ_2-10', 4);
        если I=3 то на Ml;  E3:
      конец;
      если I= 5 то на П [1];
      M2: КОД ('ПЧ _ 2- 10', 5);
      начало целый A;  на П [2]
      конец
    конец;
    M2:E2:КОД ('ПЧ_ 2-10', 6); I:=4;
    на П [1]; M3:КОД ('ПЧ_2-10', 7)
конец
```

Результатом работы программы является следующая последовательность чисел:

111, 2, 4, 3, 2, 4, 111, 2, 4, 5, 6, 111, 7

2.10. УСЛОВНЫЕ ОПЕРАТОРЫ

Условные операторы в языке АЛГЭМ могут быть 2-х типов.

Условный оператор 1-го типа имеет вид:

<условие><оператор>

Условный оператор 2-го типа имеет вид:

ко <целое без знака>**на** <метка>

Условный оператор может быть помечен. Условием называется последовательность символов, состоящая Из ограничителя **если**, отношения и ограничителя **то**. Например:

если X≥0 **то**

Отношением называется совокупность двух выражений, разделенных одним из знаков ≤ < = ≠ ≥ >. Арифметическое выражения, образующие отношение, должны быть одного типа: либо оба целые, либо оба вещественные.

Оператор, входящий в условный оператор, выполняется в том случае, когда отношение, входящее в условие, является истинным. В противном случае он пропускается, и выполнение программы продолжается с оператора, следующего за условием.

Примеры:

если X>0 **то на** P2
если K=J **то** M1:A:=A+1

Из первого примера следует, что X должен обозначать переменную типа **целый. (Е**сли бы условие имело вид X>0.0, переменная X должна была бы иметь тип **вещественный**). Переменные К и I могут быть либо целыми, либо вещественными.

При использовании вещественных выражений в отношениях следует иметь в виду, что вследствие погрешностей машинного представления вещественных чисел значение отношения может в некоторых случаях отличаться от значения, определяемого записью алгоритма. Например:

```
начало целый I;  вещественный K;   КОД ('ПЧ', 5.0);
   для I: = 1 шаг 1 до 10 цикл
   начало КОД ('ПЧ_2-10', I);
     K:=I;
     если K≤5.0 то КОД ('ПЧ', K)
   конец
 конец
```

В данном примере отношение K≤5.0 перестает быть истинным, начиная с I, равного 5, так как функция преобразования значения из типа **целый** в тип **вещественный**, используемая в операторе присваивания

K: = I,

не дает погрешностей, а значение константы 5.0, переведенное в двоичную систему счисления посредством стандартной программы перевода вещественных чисел, имеет погрешность в младшем разряде мантиссы.

Результатом работы этой программы будет такая последовательность чисел:

```
+477777777403
+000000001
+400000000001
+000000002
+400000000002
+000000003
+600000000002
+000000004
+400000000003
+000000005
+000000006
+ 000000007
+000000008
+000000009
+000000010
```

Условный оператор 2-го типа, так называемый «переход по ключу», вызывает выполнение оператора перехода, входящего в его состав, в том случае, когда из совокупности ключей, определяемой целым без знака, стоящим после символа **ко**, не включен ни один ключ. Если включен хотя бы один ключ из указанной совокупности, оператор перехода по ключу эквивалентен пустому оператору.

Ключи на пульте управления имеют следующие номера: 1,2,4, 10,20,40,100.

Пример:

ко 46 **на** M3

Переход к оператору с меткой M3 осуществляется в том случае, если на пульте управления выключены ключи с номерами

40, 10,4,2 ($46_{10} = 56_8 = 40_8 + 10_8 + 4 + 2$).

При использовании ключей на пульте управления следует иметь в виду, что рабочая программа использует ключи 20 и 100 (см.. разд. 3.9.2).

2.11. ОПЕРАТОРЫ ОСТАНОВА

Оператор останова имеет вид:

стоп <целое без знака>

Целое без знака, указанное после символа стоп, не может превышать 4095. Например:

стоп 1
стоп 578

Оператор останова вызывает останов рабочей программы с высвечиванием на сумматоре (в двоичном коде) целого без знака, использованного в данном операторе, и адреса команды, соответствующей этому оператору. При нажатии кнопки «пуск» выполнение программы продолжится со следующего оператора.

Пример:

```
начало целый массив A [1:100]; целый I;
  для I: = 1 шаг 1 до 100 цикл A [I]: = 1;
  стоп 578;
  КОД ('ПЧ_2-10', A)
конец
```

Указанная программа работает следующим образом:

после пуска 07727 (см. разд. 3.9.1) последует останов, при этом значение СчАК 07703, на сумматоре — 0000 1102 7702 ($578_{10} = 1102_8$);

после нажатия кнопки «пуск» последует печать на ТБПМ и останов, СчАК 7771 (см. разд. 3.9.1).

2.12. ОПЕРАТОРЫ ЦИКЛА

Оператор цикла служит для того, чтобы заставить входящий в его состав внутренний оператор выполняться пуль или более раз. При каждом выполнении внутреннего оператора некоторой переменной— параметру цикла — присваивается новое значение. В качестве параметра цикла допускается простая переменная, имеющая согласно описанию тип **целый** или **вещественный** и не имеющая указания вида (см. раздел 2.16).

Каждое новое значение параметра цикла задается с помощью списка цикла. В АЛГЭМе имеются два способа задания значений параметру цикла и, соответственно, два типа списка цикла.

Список цикла первого типа представляет собой список арифметических выражений. Например, Al, A2. ..., AN. Такой список цикла задает параметру цикла последовательные значения, равные значениям этих арифметических выражений. Тотчас вслед за присваиванием параметру цикла очередного значения выполняется внутренний оператор.

Список цикла второго типа называется арифметической прогрессией. Он состоит из трех арифметических выражений, разделенных символами **шаг** и до:

A1 **шаг** A2 **до** A3,

где A1, A2 и A3 — арифметические выражения.

Список цикла второго типа задает параметру цикла некоторую последовательность значений; для каждого значения в этой последовательности проверяется, является ли это значение допустимым, т. е. должен ли выполняться внутренний оператор при этом значении параметра цикла. После того, как среди значений, задаваемых параметру цикла списком цикла типа арифметической прогрессии, встретилось первое недопустимое значение, выполнение программы продолжается с оператора, являющегося преемником данного оператора цикла.

Оператор цикла состоит из заголовка цикла и следующего за ним оператора.

Заголовок имеет вид:

для <параметр цикла>: = <список цикла>

Оператор цикла в случае, когда списком цикла является список выражений, имеет вид:

для V: =A1, A2,..., AN цикл S;

Этот оператор цикла эквивалентен следующей последовательности операторов:

V:=A1; **S;** V:=A2; **S;** ...; V:=AN; S;

Оператор цикла в случае, когда списком цикла является элемент типа арифметической прогрессии, имеет вид:

для V: =A **шаг B до C цикл S;**

Такой оператор эквивалентен следующей последовательности операторов (здесь B1 и C1 того же типа, что и V):

V:=A; B1:=B**;** M1:C1**:=**C**;**

если знак ((V—C 1**)** ×B 1**)** > 0 **то на** M2; S**;** B1**: =**B; V**: =**V+B1**; на** Ml**;** M2**: . ..**

Следует обратить внимание на то, что значения выражений B и C вычисляются заново при каждом выполнении оператора S.

Простейший случай — это когда выражения A, B, C не зависят •от параметра цикла V и когда ни значения этих выражений, ни параметр цикла не изменяются при выполнении оператора S. При этом список цикла типа арифметической прогрессии действительно задает параметру цикла значения

A, A+B, A+2×B,

образующие возрастающую или убывающую арифметическую прогрессию. Допустимыми будут только те значения из этой последовательности, которые не превосходят значения C при B>0 или не меньше C при B<0, В общем же случае последовательность значений, принимаемых параметром цикла в ходе выполнения оператора цикла, не является арифметической прогрессией, поскольку значения компонентов, входящих в арифметические выражения A, B и C, а также значение самого параметра цикла могут меняться в операторе, являющемся внутренним оператором цикла.

Примеры:

для P: = 1 **шаг** 2 **до** 8

определяет числа 1, 3, 5, 7.

для P: = 1 **шаг** 10 **до** 9

определяет число 1.

для P: = 1 **шаг** 1 **до** 0

не определяет никаких чисел.

для P:=A **шаг** A—P **до** C

определяет бесконечную последовательность значений параметра цикла при любых постоянных значениях A.

для P: = 10 **шаг**—1 **до** 5

определяет числа 10, 9, 8, 7, 6, 5.

Последовательность операторов, соответствующая списку цикла типа арифметической прогрессии, содержит метки Ml и М2. Метка Ml обеспечивает многократное выполнение оператора S для различных значений, присваиваемых параметру цикла данным элементом. Метка М2 предшествует оператору, являющемуся преемником оператора цикла. Переход на эту метку означает, что очередное значение, выработанное элементом списка цикла, оказывается недопустимым.

Вход в цикл возможен только через заголовок цикла. Другими словами, если некоторый оператор перехода, не входящий в состав оператора цикла, ведет к метке, стоящей внутри оператора цикла, то результат действия такого оператора не определен.

После выхода из оператора S с помощью какого-либо оператора перехода, значение параметра цикла будет таким, каким оно было непосредственно перед выполнением оператора перехода.

Если выход из цикла вызван исчерпанием списка цикла, значение параметра цикла не определено. Например, оператор цикла

для I: = 1 **шаг** 1 **до** N **цикл**

если A [I] =B **то на** L;... L: ...

предназначен для поиска среди элементов вектора A элемента, равного B. Если такой элемент будет найден (пусть даже это будет последний элемент A[N]), то на этом работа оператора цикла прервется, и будет работать оператор, помеченный меткой L. При этом переменная I сохранит значение индекса того элемента вектора A, который оказался равным B. Если же ни один из элементов вектора A не равен B, вслед за оператором цикла будет вы. подняться непосредственно следующий за ним оператор, и значение переменной I будет не определено.

Примеры:

1. Вычислить сумму векторов A и B, заданных компонентами

A [1]**,...,** A [N] и B [1] , . . . , B [N].

Используем следующий оператор цикла:

для I:=1 **шаг** 1 **до** N **цикл C** [I]:=A [I]+B [I]

2. Вычислить скалярное произведение этих же векторов S:

S:=0**;** **для** Q**: =** 1 **шаг** 1 **до** N **цикл** S**: =** S+A [Q]×B [Q]

3. Вычислить и напечатать таблицу значений *ln*(X) для значений аргумента, заданных списком цикла:

для X:=.1, .2, .3, .5, .8, 1.0, 1.5 **цикл**

начало A**: =1n (X);** КОД ('ПЧ _2-10', A)

конец

При организации циклов с использованием вещественных выражений в списках циклов и переменных типа **вещественный** в качестве параметров циклов следует иметь в виду, что вследствие погрешностей машинного представления вещественных чисел число повторений тела цикла в некоторых случаях может отличаться от указанного в записи алгоритма на АЛГЭМе. Например:

начало вещественный A;
для A**: =**3.0 **шаг—**1.0 **до** 0.0 **цикл начало**
КОД ('ПЧ', A); КОД ('ПЧ_2-10', A) **конец**
конец

Результатом работы этой программы будет следующая последовательность чисел:

+577777777402
+000000003
+ 777777777401
+000000002
+ 777777777400
+ 000000001

2.13. СОСТАВНЫЕ ОПЕРАТОРЫ

Часто возникает необходимость объединить группу операторов в одно целое. Например, внутренний оператор условного оператора или оператора цикла по правилам синтаксиса алгоритмического языка должен представлять собой один оператор. Однако часто для описания всех необходимых действий требуется использовать несколько операторов. Их можно объединить в один составной оператор.

Составным оператором называется группа из одного или нескольких операторов, заключенная в операторные скобки **начало** и **конец.** Помещать точки с запятой перед основным символом. **конец** и после основного символа **начало** не разрешается. Составной оператор можно представить в следующем виде:

начало S1; S2; ...; SN **конец**, где SI, S, . . . , SN —

внутренние операторы составного оператора. Например:

```
начало V:=A[I];   A [I]: =A [K];
A [K] :=V конец
```

Пример:

Пользуясь условным и составным операторами, описать задачу нахождения суммы положительных и суммы отрицательных чисел последовательности

$A_0, A_1,..., A_N$.

Решение,

```
...V:=U:=J:=0;
M: если J≤N то
начало если A[J]≥0 то U:=U+A[J];
   если A [J]<0 то V:=V+A [J]; J:=J+1; на M
конец
```

2.14. БЛОКИ

Блок отличается от составного оператора тем, что в нем между символом **начало** и первым внутренним оператором размещается одно или несколько описаний, отделенных друг от друга и от операторов точками с запятой. В АЛГЭМе существуют следующие описания: описания типа, описания массивов, описания составных переменных и составных массивов и описания переключателей.

Пример:

```
начало целый J, K; вещественный W;
   для J:=l шаг 1 до M—1 цикл
   для K: = J+ 1 шаг 1 до M цикл
   начало W: = A [J, K];   A [J, K]: =A [K, J];
    A [K, J]:=W
   конец
конец ...
```

Этот блок состоит из описаний типа переменных J, K, W и одного оператора цикла с параметром J. Внутренним оператором оператора цикла служит также оператор цикла с параметром K. Внутренним оператором второго цикла является составной оператор, состоящий из трех операторов присваивания. Блок осуществляет транспонирование квадратной матрицы A порядка M.

Каждый блок вводит новую систему обозначений, действующую только внутри этого блока. Эта система обозначений охватывает .все идентификаторы (переменных, массивов, составных переменных, составных массивов, переключателей), описанные в начале этого блока, а также идентификаторы, стоящие перед внутренними операторами блока в качестве меток. Все эти идентификаторы считаются локальными в данном блоке.

Это означает, что все объекты (переменные, массивы, составные переменные, составные массивы, метки и переключатели), представленные этими идентификаторами в пределах данного блока, не существуют вне блока, а также, что любой объект, представленный одним из этих идентификаторов вне блока, не может быть использован в блоке.

Каждый идентификатор, встречающийся в программе, должен быть описан. Это относится к идентификаторам как внутренних, так и внешних величин (см. разд. 2.3). Из этого правила есть исключения. Не описываются идентификаторы стандартных функций и не описывается идентификатор оператора процедуры-кода (КОД). Кроме того, не описываются специально идентификаторы меток; считается, что появление метки перед оператором есть ее описание.

В одном блоке каждый идентификатор обозначает только один объект и поэтому может быть описан (или использован в качестве метки оператора) только один раз. Исключение составляет тот случай, когда идентификаторы используются для обозначения величин, входящих в более крупные единицы информации (см. разд. 2.16.3 и 2.16). В такой ситуации один и тот же идентификатор может быть описан в блоке многократно.

В том случае, когда идентификатор описан в одном блоке многократно (за исключением описанной выше ситуации), возможны варианты:

в процессе трансляции выдаются сообщения о семантических ошибках (см. разд. 3.6.2), после чего происходит останов;

транслятор выдает рабочую программу, однако в процессе ее работы возможны коллизии в тех случаях, когда невозможно определить, о какой величине идет речь. Например:

```
начало целый A, B, C; целый массив A. P вид 99 [1:10]; .. .
   A : = 1;...
   A [5] : = 12;...
   КОД ('ПЧ_2-10',   A, B, C ) ; . . .
конец
```

В первом и во втором операторах присваивания речь идет, соответственно, о простой переменной A и о переменной с индексами, представляющей отдельный элемент массива A. В операторе процедуры-кода в качестве фактического параметра A может с одинаковым успехом использоваться как простая переменная A, так и массив A. Такая программа, очевидно, может работать неправильно.

Локальность идентификатора в блоке приводит к тому, что внутри блока этот идентификатор может употребляться только в соответствии со своим описанием. Рассмотрим два блока, вложенных один в другой, т. е. такие два блока, из которых второй блок является внутренним оператором первого (внешнего) блока. Тогда для любого идентификатора, описание которого имеет силу во внешнем блоке, могут представиться два случая. Этот идентификатор может быть не описан во внутреннем блоке; тогда он сохраняет силу, т. е. используется в том же смысле, что и во внешнем блоке; если же этот идентификатор описан снова во внутреннем блоке, его первое описание (во внешнем блоке) временно теряет силу, и на время выполнения внутреннего блока этим

идентификатором обозначается другой объект. После выхода из внутреннего блока восстанавливается смысл идентификатора, имевшего силу до входа во внутренний блок.

Приведем пример программы с вложенными блоками:

```
Р: начало вещественный А, В;
   А: = 1;  В:=2;
   Н: начало вещественный А, С;
      С:=В;   А:=3;
      КОД ('ПЧ_2-10',  А, В, С)
   конец;
   КОД ('ПЧ_2-10', А, в)
конец
```

Эта программа печатает числа 3, 2, 2, 1, 2. Первым оператором блока Н вместо С: =В нельзя написать С: =А, так как значение переменной А, локальной в блоке Н, не определено в момент входа в блок Н.

Случай, когда блоки В1 и В2 являются независимыми под блоками одного и того же блока В,— относительно прост. Операторы блока В1 не могут использовать локальные переменные блока В2 и наоборот. Значения локальных переменных блока В1 не определены в момент работы блока В2 и наоборот; следовательно, для локальных переменных блоков. В1 и В2 могут быть использованы одни и те же ячейки памяти.

Другими словами, одинаковые идентификаторы, описанные или примененные в качестве метки оператора в разных блоках, изображают различные величины. Из этого, в частности, следует, что вход в блок возможен только через его начало, так как все внутренние метки блока локальны в нем; они не имеют смысла вне блока и не могут употребляться во внешнем блоке для переходов во внутренний блок.

Пример:

```
начало целый А; …
   Р :начало целый  В; . . . ; М : А : = 3; …
   конец; на Р; ...; на М; . ..
конец
```

Здесь оператор на Р обеспечит передачу управления в начало блока Р, а метка М в операторе перехода на М не имеет смысла, так как метка М блока Р локальна в нем и не определена вне его.

Принцип локализации идентификатора с помощью описаний можно выразить иначе. Для каждого идентификатора, встретившегося в каком-либо участке программы, соответствующее ему описание следует искать в начале наименьшего блока, охватывающего этот участок. Если в этом блоке описание данного' идентификатора не встретится, то следует перейти к минимальному блоку, охватывающему этот блок. Если и в нем не найдется нужное описание, то следует обратиться к наименьшему блоку, содержащему второй блок, и т. д. Аналогичный процесс применяется при поиске метки, к которой ведет оператор перехода. Свойство локальности меток по существу не отличается от локальности других идентификаторов.

Например, в приведенном выше примере описание идентификатора М (использованного в операторе перехода на М) следует искать во внешнем блоке, так как он является наименьшим (и единственным в данном случае) блоком, содержащим оператор на М. Описанием метки является ее появление с двоеточием в блоке; следовательно, в данном случае М — неописанный идентификатор. Метка М описана в блоке Р, не являющемся охватывающим по отношению к оператору на М.

Участок программы, на котором имеет силу описание некоторого идентификатора, называется областью действия этого идентификатора.

Областью действия идентификатора является совокупность операторов блока, в начале которого помещено его описание, за исключением тех внутренних блоков (вместе с их подблоками), в которых определены и описаны такие же идентификаторы.

Принцип локальности позволяет для любых величин, используемых только внутри некоторого блока, вводить любые обозначения (идентификаторы), не заботясь о том, употреблялись ли такие обозначения во внешних блоках. При этом, разумеется, для всех величин, введенных во внешних блоках и используемых в данном блоке, сохраняются прежние обозначения.

Пример:

```
начало целый А; А:=3;
   М: начало целый В, Х; В:=2; Х:=А+В
   конец
конец
```

Здесь идентификатор А нельзя вторично описать в блоке М, если мы хотим сохранить его значение в этом блоке.

Описания, располагающиеся в начале блока, сообщают о существовании в пределах блока тех или иных объектов. По описаниям переменных типа **строчный,** массивов, составных переменных и составных массивов вычисляются значения границ индексов и длины соответствующих переменных. Эти значения вычисляются заново при каждом входе в блок (через его начало) и не меняются во время выполнения внутренних операторов блока.

В момент входа в блок значения его локальных величин еще не определены. Поэтому в выражениях для границ индексов-(см. разд. 2.16.2) разрешается употреблять только не локальные идентификаторы.

Описания переключателей вступают в действие только при обращении к ним посредством встречающихся в блоке .указателей переключателей.

После (вычисления границ индексов в описаниях массивов и составных массивов и выполнения некоторых других вспомогательных действий начинает выполняться первый внутренний оператор, стоящий непосредственно вслед за описаниями. Выход из блока может произойти как вследствие того, что выполнен последний внутренний оператор блока, так и в результате выполнения оператора перехода, ведущего к метке, расположенной вне блока. В первом случае мы возвращаемся в блок, который непосредственно охватывает

только что выполненный блок. Во втором случае метка, к которой осуществлен переход, может находиться в блоке еще более низкого уровня, т. е. мы можем выйти сразу из нескольких содержащих друг друга блоков.

2.15. ПРИМЕЧАНИЯ

В текст программы, записанной на АЛГЭМе разрешается вставлять так называемые примечания. Примечания бывают двух типов.

1). В первом случае примечание всегда начинается с основного символа примечание, за которым следует любой текст, не содержащий символа «;». Такие примечания помещаются после основного символа начало или после точки с запятой. Примечание всегда должно заканчиваться точкой с запятой. Перед символом конец примечание первого вида помещать нельзя.

Пример:

```
начало целый массив A [1:100];
  примечание_ А-МАССИВ _РЕЗУЛЬТАТОВ;
  целый В; примечание _В-ПАРАМЕТР _ ЦИКЛА;
  для В:=,1 шаг 1 до 100 цикл А [В] :=В;
  КОД ('ПЧ_2-10', А);
  примечание _  ПОЛУЧЕН _  И _НАПЕЧАТАН _  МАССИВ _ А;
  стоп 1
конец
```

2). Во втором случае примечание всегда помещается после основного символа **конец**. Текст примечания не должен, содержать символов **конец** и « ; ». Конец примечания определяется ближайшим символом **конец** или « ; ».

Пример:

```
начало целый I,  J;  массив A [1:10,  1:10];
для  I: = 1 шаг 1 до 10 цикл
для J: = 1 шаг 1 до 10 цикл
начало A [I, J]: =I —J;.
   КОД ('ПЧ_ 2- 10',  A [I, J])
  конец _ КОНЕЦ_ ЦИКЛОВ_ ПО_  I_  И_ ПО_ J_
конец _ ПРОГРАММЫ
```

Оба вида примечаний не оказывают никакого влияния на программу (при трансляции они удаляются), а используются программистом для придания программе удобочитаемого вида.

2.16. ОПИСАНИЯ

Как указывалось выше, описания служат для того, чтобы определить некоторые свойства величин, используемых в программе, и связать эти величины с идентификаторами.

Описания (одно или насколько) помещаются в начале блока между символом начало и первым внутренним оператором блока. Описания отделяются друг от друга и от операторов точками с запятой.

Существуют следующие описания: описания типа, описания 'массивов, описания составных переменных, описания составных массивов и описания переключателей. Описания переключателей были рассмотрены выше (см. разд. 2.9). Ниже будут рассмотрены остальные разновидности описаний.

2.16.1. Описания типа

Описание типа служит для указания того, что некоторые идентификаторы представляют простые переменные данного типа. Различают следующие типы простых переменных:

целый вещественный строчный

Переменные, которым по описанию приписан тип **целый,** могут принимать положительные и отрицательные целые значения, включая нуль, т. с. значениями переменных типа **целый** могут быть только целые числа (в этом случае нуль изображается как 0, а в коде машины «Минск-22» имеет вид + 0000 0000 0000).

Переменные, которым по описанию приписан тип **вещественный,** могут принимать положительные и отрицательные вещественные значения, включая нуль, т. е. значениями переменных типа **вещественный** могут быть только вещественные числа (в этом случае нуль изображается как 0.0, а в коде машины «Минск-22» имеет вид+000000000177). .

Переменные, которым согласно описанию приписан тип **строчный,** принимают значения, являющиеся открытыми строками (см. разд. 2.5).

Все переменные типа **строчный** должны иметь в описании указание вида (см. разд. 2.6.1). Переменные типа **целый** и **вещественный** могут иметь в описании указание вида (см. разд. 2.6.1). Переменные, имеющие в описании указание вида, рассматриваются как величины, значения которых имеют фиксированную длину и фиксированный состав символов в заданных позициях. Указание вида характеризует только ту переменную, за идентификатором которой оно непосредственно следует в описании.

Все переменные, перечисленные в одном описании типа, имеют один и тот же тип.

Примеры:

```
целый А,  В,  А1
вещественный С, Д вид 9.9, М
строчный Е вид С (10)
```

В первом описании типа сообщается о том. что идентификаторы А, В и А1 представляют в соответствующем блоке простые переменные типа **целый.**

Во втором примере вещественная переменная, обозначенная идентификаторам Д, имеет вид 9.9, тогда как

вещественные переменные, названные идентификаторами С и М, не имеют указаний вида.

Указание вида переменной состоит из основного символа **вид** и формата, состоящего в общем случае из указателей символов 9 1.$_{10}$+—С и повторителей (структуру формата и действия указателей символов см. в разд. 2.6.1—2.6.4).

Примеры:

вещественный А **вид** +99.9, В **вид** 9.9$_{10}$+9
строчный Д **вид** С (15)
целый Е **вид** 999, М **вид** 9 (5)

2.16.2. Описания массивов

Массив — это совокупность величин программы, обозначенных одним и тем же идентификатором (например, Р). Компонентами массива являются переменные с индексами. Например,

P [1], P [2], P [3], P [4], P [5] …

Все компоненты одного массива должны принимать значения одного типа. Для описания массива используются описатели:

массив **целый массив** **строчный массив**,

за которыми следуют списки идентификаторов и списки граничных пар. Например:

целый массив А [1:5, 1:10]
массив В **вид** 9.9, С [4: К]
строчный массив Д **вид** С (5), Е **вид** С (3), М **вид** С (10) [1:6]
массив А1, А2 [1:50], В1 **вид** 9.9, В2 [1:N]

Конструкция, заключенная в квадратные скобки, называется списком граничных пар. Список граничных пар относится к идентификаторам, перечисленным слева от него, вплоть до следующего списка граничных пар или основного символа **массив**. Так, в последнем примере список граничных пар [1 :50] относится к идентификаторам массивов Al и А2, а список граничных пар [1 : N] — к идентификаторам массивов В1 и В2.

Если перед символом **массив** отсутствует описатель, такой массив понимается как имеющий тип **вещественный**.

Список граничных пар показывает:

сколько индексов должна иметь запись соответствующей переменной с индексами, т. е. определяет размерность массивов;

в каких пределах должно изменяться значение соответствующего индексного выражения (см. разд. 2.4.2).

Граничные пары отделяются друг от друга запятыми. Каждая граничная пара состоит из двух выражений, разделенных двоеточием:

целый массив М [К:L, 1:5, —3: —1]

Первое (левое) выражение каждой граничной пары дает нижнюю границу, а второе (правое) выражение задает верхнюю границу значения соответствующего индексного выражения. В качестве выражений в граничных парах (не смешивать с индексными выражениями!) могут быть использованы:

целые числа без знака и целые числа со знаком «минус»;

простые переменные типа **целый**, не имеющие, согласно описанию, указаний вида. Кроме того, эти переменные не должны быть локальными в блоке, для которого (имеет силу данное описание массивов. Отсюда следует, что в самом .внешнем блоке программы могут быть описаны лишь массивы с постоянными границами.

Пример:

```
начало целый массив А [1:100,0:20];
  целый К, М, L; ...
  М:=5; ...; К: = l;  L : = К ×М ; …
  начало массив В [К: L]; . ..
  конец;...
конец
```

Пример неправильной программы:

```
начало целый массив А [1:N]; целый N, I;
  КОД ('ВВОДЛ _ 10-2'. N);
  для I: = 1 шаг 1 до N цикл А [I]: =1;
  КОД ('ПЧ _ 2-10', А)
конец
```

Эту программу следует записать, например, так:

```
начало целый N, I;  КОД ('ВВОДЛ _ 10—2', N);
начало целый массив А [1:N];
    для I: = 1 шаг 1 до N цикл А [I] : = I;
    КОД ('ПЧ  2—10', А)
  конец
конец
```

Значения границ в описаниях массивов определяются при входе в блок, в котором описан данный массив. Массив определен только в том случае, когда значения всех верхних границ не меньше значений соответствующих нижних границ.

В описании массивов могут быть использованы указания вида, задающие длины и состав символов значений компонентов массива. Указание вида распространяется на идентификатор массива, стоящий непосредственно перед таким указанием. Например:

массив А, В **вид** 99.9, С [-5:0], Д [1:3];
строчный массив Е **вид** С (10) [1:10]

Здесь массив В имеет указание вида. Массивы А, В и С имеют одинаковые размерности и пределы изменения значений индексов [—5 : 0].

Строчные массивы всегда должны иметь в описаниях указания вида. Массивы типа **целый** или **вещественный** могут иметь, а могут и не иметь в описании указания вида.

Для того, чтобы указать какой-либо компонент массива, используется переменная с индексами. Запись переменной с индексами состоит из идентификатора массива, за которым следует заключенный в квадратные скобки' список индексов. Список индексов состоит из одного или нескольких арифметических (целых или вещественных) выражений, разделенных запятыми.

Число этих арифметических выражений, называемых индексными выражениями, должно быть равно числу граничных пар в соответствующем описании массивов (см. также разд. 2.4.2).

Для того, чтобы обеспечить возможность обмена информацией между внешней памятью (магнитными лентами) и оперативной памятью ЭВМ, используются внешние массивы. Идентификатор внешнего массива всегда начинается с буквы Я, за которой следует истинный начальный адрес внешнего массива на МЛ—восемь восьмеричных цифр. Например:

массив Я02004000 [1:К]

Первые две цифры дают, соответственно, номер шкафа и ЛПМ, остальные шесть цифр определяют начальный адрес на магнитной ленте. Тип и указание вида в описании внешнего массива используются обычным образом. Идентификаторы внешних массивов могут перечисляться в общем списке, наряду с идентификаторами внутренних массивов. Например:

массив А, Я01005000 **вид** 99.99, В [1 : 1000], Я10007000, С [1:50]

Идентификатор внешнего массива может быть использован только в качестве фактического параметра оператора процедуры-кода со спецификациями 'ЗАПИСЬ' и 'ЧТЕНИЕ' (см. разд. 2.17).
Обращения к отдельным элементам внешних массивов не допускаются.

Пример:

```
начало целый К; ...
    К: =200';...
    начало массив А, А1, Я02001000 [1:К];…
    КОД ('ЗАПИСЬ', А, Я02001000); …
    КОД ('ЧТЕНИЕ', Я0200Г000, А1); ...
    конец; ...
конец
```

В этой программе сначала производится перепись массива А из МОЗУ на МЛ, а затем чтение с МЛ в МОЗУ на место, отведенное для массива А1. Длина массива равна 200_{10} ячейкам.

2.16.3. Описания составных переменных и внешних составных переменных

Для задания структур различных документов, алгоритмов их обработки и организации выдачи их на АЦПУ, а также для обмена информацией между оперативной памятью и магнитными лентами служат описания составных величин: составных переменных и составных массивов, а также описания внешних составных переменных и внешних составных массивов.

Описание составной переменной начинается с основного символа-скобки **составной,** за которой следует идентификатор, именующий эту составную переменную. Идентификатор составной переменной отделяется точкой с запятой от описаний остальных переменных и массивов (которые, в свою очередь, могут быть элементарными или составными), входящих в состав данной составной переменной. Этим же описанием задается последовательность расположения величин в составной переменной и уровни их иерархии. Заканчивается описание составной переменной основным символом-скобкой **уровень.** Перед символом **уровень** точка с запятой не ставится. Описания, находящиеся внутри описания составной переменной, отделяются друг от друга точками с запятой:

А								
Б	В	В1		Н[1]	Н[2])	Н[3]	Н[4]	Н[5]
		Д	Д1					
ИВАНОВ	125	13	14	2,5	3,5	4,5	6,5	7,5

Описание этого документа задается в виде описания составной переменной:

```
составной А;
  строчный Б вид С (6);
  целый В;
  составной В 1;
    целый Д вид 99, Д1 вид 99
    уровень;
    массив Н вид 9.9 [1:5]
уровень
```

Составная переменная не имеет типа и вида, так как в общем случае состоит из величин различных типов и видов. Только элементарная величина (переменная или массив) может иметь тип и вид, что и указывается в

описании на соответствующем уровне. Поскольку элементами составных переменных могут быть массивы, то к ним относится все выше сказанное об описании массивов.

Обращение к элементу составной переменной начинается с элементарной переменной и содержит в качестве уточнений все промежуточные уровни иерархии, включая самый внешний (см. также разд. 2.4.2). Например, обращения к элементам составной переменной, описанной выше, будут записаны так:

Д. В1.А (значение равно 13)
Н [3]. А (значение равно 4. 5)

Уточнения, т. е. ссылки на более крупные единицы информации, в состав которых входит данная элементарная переменная, отделяются друг от друга и от элементарной переменной точками (см. разд. 2.4.2).

Для обозначения элементарных переменных и составных переменных и массивов, находящихся согласно описанию в различных составных переменных и составных массивах, разрешается использовать одинаковые идентификаторы даже в том случае, если эти описания находятся в одном и том же блоке. Например:

```
начало
    составной ОТЧЕТ;
        целый СПРАВКА; массив СПИСОК [1:50];
        вещественный ИТОГ
    уровень;
    составной ОТЧЕТ1;
        целый СПРАВКА;   массив СПИСОК [1:50];
        вещественный ИТОГ
    уровень;
    КОД ('ВВОДЛ', ОТЧЕТ); . ..
    если СПРАВКА.  ОТЧЕТ = СПРАВКА.  ОТЧЕТ1  то  на  Ml;
    ИТОГ.  ОТЧЕТ:=ИТОГ.  ОТЧЕТ 1;
    Ml:
конец
```

Для обмена информацией между оперативной памятью ЭВМ и магнитными лентами введено описание внешней составной переменной. Оно имеет, например, следующий вид:

```
составной Я03004000; 5×К+М,  3+А уровень
```

Идентификатор внешней составной переменной начинается с буквы Я, за которой следует восемь восьмеричных цифр — ленточный адрес (см. разд. 2.16.2). После точки с запятой следует два целых арифметических выражения, разделенных запятой. Значение первого выражения представляет собой сдвиг относительно начального ленточного адреса (заданного идентификатором внешней составной переменной), а значение второго—задает длину (количество ячеек) соответствующей внешней составной переменной. Переменные, входящие в состав этих двух арифметических выражений, не должны быть локальными в блоке, для которого имеет силу описание данной внешней составной переменной.

Идентификаторы внешних составных переменных могут использоваться только в качестве фактических параметров оператора процедуры-кода со спецификациями 'ЗАПИСЬ' и 'ЧТЕНИЕ' (см. разд. 2.17).

Пример:

```
начало массив А, А1 [1:100];
  составной Я01001000;  500, 300 уровень;
  КОД ('ВВОДЛ _ 10-2', А);
  КОД ('ЗАПИСЬ',  А, Я01001000);  КОД ('ЧТЕНИЕ', Я01001000, А1),
конец
```

В программе выполняется запись на магнитную ленту массива А, причем записываются 300_{10} ячеек, начиная с ленточного адреса 01001764_8 ($01001000_8 + 764_8$). Далее выполняется чтение внешней составной переменной с магнитной ленты (с того же адреса) в оперативную память на место, отведенное под массив А1, причем читается 100_{10} ячеек.

Отметим, что в процедурах-кодах со спецификациями 'ЗАПИСЬ' и 'ЧТЕНИЕ' фактические параметры записываются парами, в которых первый параметр является отправителем, а второй— получателем. Количество ячеек, участвующих в обмене, определяется получателем (см. также разд. 2.17).

2.16.4. Описания составных массивов и внешних составных массивов

Выше приводилось описание составной переменной и внешней составной переменной. Составные массивы и внешние составные массивы—это массивы, состоящие из одинаковых по структуре составных переменных, соответственно. В описании составного массива внешнего составного массива вслед за идентификатором массива следует список граничных пар, заключенный в квадратные скобки. Например:

```
составной массив А [1:300];
  строчный Б вид С (6);
  целый В;
  составной В1;
     целый Д вид 99, Д1 вид 99
  уровень;
  массив Н вид 9.9 [1:5]
уровень
```

Этот составной массив состоит из одинаковых по структуре составных переменных. Количество таких (составных переменных в данном случае равно 300.

К составным массивам относится все сказанное выше о количестве и составе граничных пар и о соответствии между граничными ларами в описании составного массива и индексными выражениями в уточнениях. Например, обращение к элементу такого составного массива может иметь вид:

Д. Bl. A [25]

H [3], A [76]

В первом случае имеется в виду то значение переменной Д, которое находится в 25-м документе; во втором случае — то значение третьего элемента массива Н, которое находится в 76-м документе (всего имеем, согласно описанию, 300 документов одинаковой структуры).

Описание внешнего составного массива может иметь, например, такой вид:

составной массив Я01001000 [1:M]; A+B, C+Д **уровень**

Здесь два арифметических выражения, стоящих после точки с запятой, имеют тот же смысл, что и в описании внешней составной переменной. Число граничных пар в описании внешнего составного массива может быть любым. Для данного примера начальный ленточный адрес K-го элемента этого массива Я01001000 вычисляется следующим образом:

$$01001000_8 + (A+B) + (K-1) \times (C+Д)$$

Пример

Пусть имеется массив целых чисел А [1:N]. Из этого массива требуется выбрать три группы чисел таким образом, чтобы в первую группу попали числа, лежащие в интервале [$Д_1$, $Д_2$), во вторую — числа в интервале [$Д_2$, Дз). и в третью — в интервале [$Д_3$, $Д_4$).

Полученные группы чисел записать на магнитную ленту вплотную друг за другом и напечатать.

```
начало целый N;  КОД ('ВВОДЛ_ 10—2', N);
   примечание ВЫПОЛНЯЕТСЯ_ ВВОД _С_ ПЕРЕВОДОМ _В_ ДВОИЧНУЮ _ СИСТЕМУ _ СЧИСЛЕНИЯ _ ЗНАЧЕНИЯ _ N —
КОЛИЧЕСТВА _ ЧИСЕЛ _ В _ МАССИВЕ;
   начало  целый  массив A, B  [l:N], Д [1:4];
   целый  I, J, K, L, НП, ВП;
   КОД  ('ВВОДЛ_ 10-2', А, Д);
   примечание ВЫПОЛНЯЕТСЯ_ ВВОД _ С _ ПЕРЕВОДОМ _ В _ ДВОИЧНУЮ _ СИСТЕМУ _ СЧИСЛЕНИЯ _ МАССИВА _ А _ И _
МАССИВА _ Д — ЗНАЧЕНИЙ _ НИЖНИХ - НП _ И _ ВЕРХНИХ — ВП_ ПРЕДЕЛОВ_ КАЖДОЙ _ ИЗ _ ТРЕХ _ ГРУПП _;
   L:=0;
   для K: = 1 шаг 1 до 3 цикл
   начало J:=0;  НП:=Д[К];  ВП:=Д [К+1];
      для I: = 1 шаг 1 до N цикл
      начало если НП≤А [I] то если А [I]<ВП то
      начало J:=J+1;  B [J]:=A [I] конец
   конец ВНУТРЕННЕГО _ ЦИКЛА _ ПО I , В _ КОТОРОМ _ ИЗ _ МАССИВА _ А _ ВЫБИРАЮТСЯ _ ЧИСЛА, ЛЕЖАЩИЕ
_В _ ЗАДАННЫХ _ ПРЕДЕЛАХ. ПОЛУЧЕНА _ ГРУППА _ ЧИСЕЛ _ ;
   начало составной Я01000000;  L, J уровень;
      целый K;
      КОД ('ЗАПИСЬ',  В, Я01000000);
      примечание ВЫПОЛНЕНА _ ЗАПИСЬ _ ГРУППЫ_ ЧИСЕЛ _НА _ МЛ. ДЛИНА _ ГРУППЫ _ ОПРЕДЕЛЯЕТСЯ _ ЗНАЧЕНИЕМ
_ J _ ;
      L:=L+J;

      примечание ЗНАЧЕНИЕ _ L _ ЯВЛЯЕТСЯ_ СДВИГОМ _ ОТНОСИТЕЛЬНО_ ЗАДАННОГО_ НАЧАЛЬНОГО _ ЛЕНТОЧНОГО
_АДРЕСА. ВЫПОЛНЕНО _ УВЕЛИЧЕНИЕ _ ЗНАЧЕНИЯ _ L_ НА _ ДЛИНУ _ ЗАПИСАННОЙ _ ГРУППЫ_ ЧИСЕЛ _ ДЛЯ
ТОГО, ЧТОБЫ_ СЛЕДУЮЩАЯ_ ГРУППА _ ЧИСЕЛ _ БЫЛА _ ЗАПИСАНА _ НА_ МЛ _ ВПЛОТНУЮ _ ЗА _ ПРЕДЫДУЩЕЙ.
ДАЛЕЕ _ ВЫПОЛНЯЕТСЯ _ ПЕЧАТЬ _ ГРУППЫ _ ЧИСЕЛ;

      для K: = 1 шаг 1 до J цикл
      КОД ('ПЧ_ 2-10' В, [К])
      конец ВНУТРЕННЕГО _ БЛОКА, ВЫПОЛНЯЮЩЕГО_ ЗАПИСЬ _ НА _ МЛ _ И _ ПЕЧАТЬ _ ГРУППЫ _ ЧИСЕЛ.
   конец ЦИКЛА ПО К, ОБРАЗУЮЩЕГО ВСЕ ГРУППЫ ЧИСЕЛ.
конец
конец ПРОГРАММЫ.
```

Массив Д [1 : 4] может, например, иметь вид:

```
Д [1] 0
Д [2] 3
Д [3] 10
Д [4] 100
```

Пусть N=15 и пусть исходный массив А имеет следующий вид: 1, 2, 0, 1, 0, 2, 55, 4, 105, 6, 25, 200, 8, 5, 1. Тогда результатом программы будет следующая последовательность чисел:

```
+000000001
+000000002
+000000000
+000000001
+000000000
+000000002
+000000001
+000000004
+000000006
+000000008
+000000005
+000000055
```

+000000025

2.17. ОПЕРАТОРЫ ПРОЦЕДУР-КОДОВ

Оператор процедуры-кода в языке АЛГЭМ имеет следующую структуру:

КОД (<спецификация процедуры-кода>, <список фактических параметров>)

Отдельные элементы (фактические параметры) списка фактических параметров отделяются друг от друга запятыми. Число фактических параметров в списке может быть любым. Например:.

КОД ('ПЧ_10',А, В, Д)

В этом операторе процедуры-кода задано три фактических параметра.
В качестве фактических параметров могут использоваться:
число без знака (целое или вещественное);
переменная;
строка;
групповой параметр.
Групповой параметр—это одна из следующих конструкций:
идентификатор массива;
идентификатор составной переменной;
идентификатор составного массива;
идентификатор составного массива со списком индексов;
одна из приведенных выше конструкций группового параметра с уточнениями;
идентификатор внешнего массива;
идентификатор внешней составной переменной;
идентификатор внешнего составного массива;
идентификатор внешнего составного массива со списком индексов.-
В АЛГЭМ имеются следующие спецификации оператора процедуры-кода (спецификация всегда задается в кавычках):

1). Спецификация перевода чисел из одной системы счисления в другую:

'2-10'
'10-2'

2). Спецификация выдачи числовых данных на ТБПМ:

'ПЧ'
'ПЧ _ 2-10'
'ПЧ_ 10'

3). Спецификация выдачи текстовой информации на АЦПУ-128:

'ПЧ _ АЦПУ'

4). Спецификации ввода данных с перфоленты:

'ВВОДЛ'
'ВВОДЛ_ 10-2'
'ВВОДЛ _ С

5). Спецификация вывода данных на перфоленту:

'ПРЛ'
'ПРЛ _ 10'
'ПРЛ _ 2— 10'
'ПРЛ _С'

6). Спецификации ввода данных с перфокарт:

'ВВОДК'
'ВВОДК _ 10'
'ВВОДК _ 10-2'
'ВВОДК _ С'

7). Спецификации вывода данных на перфокарты:

'ПРК'
'ПРК _ 10'
'ПРК _ 2-10'
'ПРК _ С'

8) Спецификация обмена информацией с магнитными лентами:

'ЗАПИСЬ'
'ЧТЕНИЕ'

9). Спецификация деления целых чисел:

‘Д'

10). Спецификация, используемая, при сегментации программы:

'ВЫХОД'

11). Спецификации, используемые при обращении к стандартным программам, составленным пользователями в кодах ЭВМ «Минск-22» и включенным в БСП согласно инструкции по включению в БСП новых процедур-кодов (см. разд. 3.10.2):

'<адрес>',

где <адрес > представляет собой последовательность из восьми восьмеричных цифр. Например:

'00000161'
'00000157'

Процедуры-коды в трансляторе СТ-3 оформлены в виде стандартных программ, а оператор процедуры-кода реализуется в составленной транслятором рабочей программе в виде обращения к соответствующей стандартной программе, номер которой определяется спецификацией, использованной в операторе.

Существует основное ограничение, накладываемое на фактические параметры, используемые в операторе процедуры-кода, а именно: форма представления значений соответствующих фактических параметров должна соответствовать спецификации оператора процедуры-кода.

2.17.1. Формы представления информации

Переменным различных типов **(целый, вещественный, строчный)** соответствуют различные формы представления их значений.

Значения переменных типа **целый,** не имеющих в описании указания вида, представляются записью в двоичном или двоично-десятичном коде в системе с фиксированной запятой. Такие значения записываются в конце ячейки (правых ее разрядах). Например, целое число 253 записывается в двоичном коде следующим образом:

+000 000 000 000 000 000 000 000 000 011 111 101

или в восьмеричной системе:

+00000000375,

а в двоично-десятичном коде в виде:

+0000 0000 0000 0000 0000 0000 0010 0101 0011

или в десятичной системе:

+0001000253

Значения переменных типа **вещественный,** не имеющих в описании указания вида, представляются записью в двоичном или двоично-десятичном коде в системе с плавающей запятой. В двоичном коде такие значении всегда представляются следующим образом: шесть правых разрядов ячейки дают значение порядка, седьмой разряд справа — знак порядка, остальные 28 разрядов занимает мантисса; в знаковом разряде помещается знак числа. Например, число +15.2 будет представлено в двоичном коде следующим образом:

+ 111 100 110 011 001 100 110 011 001 100 000 100

или ,в восьмеричной системе

+ 746314631404

Значение переменной типа **вещественный** в двоично-десятичном коде имеет два варианта представления в зависимости от положения ключа 20 на пульте управления (см. разд. 3.5.3, 3.9.2):

с порядком и мантиссой (ключ 20 включен),

с десятичной точкой (ключ 20 отключен).

Число +15.2 в первом варианте запишется в следующем виде:

+0001 0101 0010 0000 0000 0000 0000 0000 0010

или в десятичной системе:

+ 1520000+02

Здесь две последние десятичные цифры и знак перед ними представляют порядок, а первые семь десятичных цифр и знак перед ними—мантиссу.

Во втором варианте представления значений переменных типа **вещественный** целая часть от дробной отделяется десятичной точкой (код 1110 или 16**).**

Значение переменной записывается с правого конца ячейки. Число +15.2 запишется в следующем виде:

+0000 0000 0000 0000 0000 0001 0101 1110 0010

или в десятичной системе:

+0000015.2

Если простые переменные типа **целый** или **вещественный** имеют в описании указание вида, то значения их располагаются с левого конца ячейки в двоично-десятичном коде (если указание вида содержит указатели символов „9") или в двоичном коде (если указание вида содержит указатели символов „1"). Например:

начало целый А вид 999; **вещественный В вид** 99.9;
.., А: =253; **В: =** 15.2; **...**
конец

Значения А и В в десятичной системе в ячейках будут представлены следующим образом:

A:+ 253000000

B: + 15.200000

Точность представления значений типа **вещественный** не более семи десятичных значащих цифр.

Значения отдельных элементов массивов (т. е. переменных **с** индексами) типа **целый** или **вещественный,** не имеющих в описании указания вида, записываются каждое в отдельной ячейке. Например:

```
начало массив M [1:4]; ...
  M [1]: = 15.2; M [2]: = 0.0; M [3]: = -0.5;
  M [4]: = -16.0;    ... КОД ('2-10', M);   …
конец
```

Пусть, например, в соответствии с распределением памяти элементы этого массива располагаются, начиная с ячейки 12000. Тогда получим следующее размещение чисел в последовательных ячейках (при условии, что ключ 20 отключен)*:

```
12000) +0000015.2
    1)+00000000.
    2)-0000000.5
    3)-000000016
```

То есть количество ячеек, отводимых под массив, в данном случае соответствует числу элементов этого массива.

Значения отдельных элементов массивов типа **целый** и **вещественный,** имеющих в описании указание вида,- записываются в ячейках одно за другим в соответствии с этим указанием вида. Знаковый разряд ячейки не используется. В ячейке всего может быть помещено 9 десятичных символов или 36 двоичных символов. Если бы элементы приведенного выше массива имели указание вида +99.9, их последовательное расположение в ячейках оперативной памяти было бы следующим:

```
l2000)+ +15.2+00.
    1)+0-00.5-16
    2)+.00000000,
```

т. е. количество ячеек, отводимых под массив, в данном случае равно числу символов в указании вида, умноженному на число элементов массива и деленному на 9 (в данном случае 3 ячейки). Аналогично рассчитывается количество ячеек, отводимых под массив, если в указании вида используется указатель «'1».

Значения переменных типа **строчный** представлены в памяти машины в семиразрядных кодах АЦПУ-128. Каждой позиции значения переменной соответствуют семь двоичных разрядов. Следовательно, в ячейке помещается пять символов (знаковый и первый разряды не используются). Например:

```
начало строчный A вид C (3);...
  A: ='253';...
конец
```

В ячейках МОЗУ значение А будет представлено следующим образом:

```
+0 0000010  0000101  0000011  0000000  0000000
```

Как известно, составная переменная и составной массив не имеют типа, так как в общем случае состоят из элементарных величин различных типов. Если составная переменная или составной массив (являющиеся групповыми параметрами) используются в качестве фактических параметров оператора процедуры-кода со спецификациями, предполагающими использование числовой информации, они воспринимаются как имеющие тип вещественный.

Пример:

```
начало составной C;
  вещественный A;
  массив B [1:50], Д [1:20, 1:70]  уровень;  ...
  КОД  ('ПЧ_  2-10',C); ...
конец
```

Такая запись допустима, так как все элементы составной переменной С имеют один и тот же тип — **вещественный**, причем каждое число, входящее в совокупность С, занимает одну ячейку (подпрограммы перевода, используемые в трансляторе, осуществляют поячеечный перевод).

Другой пример:

```
начало составной C;
   вещественный A;
   массив B [1:50];
   целый массив Д [1:20,  1:70] уровень;  ...
   КОД ('ПЧ_2-10',  A.C,  Д.C);…
конец
```

* Целые числа типа **вещественный** представлены в ячейках без мантиссы, а вещественный нуль в крайней правой позиции имеет десятичную точку.

Таблица семиразрядных кодов АЦПУ-128

Символ	Код	Символ	Код
0	0000000	З	0100111
1	0000001	И	0101000
2	0000010	Й	0101001
3	0000011	К	0101010
4	0000100	Л	0101011
5	0000101	М	0101100
6	0000110	Н	0101101
7	0000111	О	0101110
8	0001000	П	0101111
9	0001001	Р	0110000
+	0001010	С	0110001
−	0001011	Т	0110010
/	0001100	У	0110011
,	0001101	Ф	0110100
.	0001110	Х	0110101
	0001111	Ц	0110110
	0010000	Ч	0110111
(	0010001	Ш	0111000
)	0010010	Щ	0111001
× Умножение	0010011	Ы	0111010
= Зн. равенства	0010100	Ь	0111011
	0010101	Э	0111100
		Ю	0111101
;	0010110	Я	0111110
[	0010111	D	1100100
]	0011000	F	1110100
*	0011001	G	1100011
' Кавычка	0011010	I	1101000
, Кавычка	0011011	J	1101001
≠	0011100	L	1101011
<	0011101	N	1101101
>	0011110	Q	1111110
:	0011111	R	1110000
А	0100000	S	1110001
Б	0100001	U	1110011
В	0100010	V	1100110
Г	0100011	W	1100010
Д	0100100	Z	1100111
Е	0100101	— Надчёркивание	0111111
Ж	0100110		

Если различные элементы составной переменной имеют по описанию разные типы, следует выполнять поэлементный перевод чисел из одной системы в другую. Если использовать оператор процедуры-кода в таком виде, как он записан в первом примере, все значения элементарных величин, входящих в составную переменную С, будут восприняты как вещественные числа, и перевод будет осуществлен неправильно.

Имеется еще одно общее ограничение, относящееся ко всем спецификациям оператора процедуры-кода, кроме спецификации '<адрес>', а именно: в качестве фактических параметров запрещается использовать переменные типа целый или вещественный, имеющие по описанию указание вида, если они являются элементами массивов или составных переменных.

Пример:

```
начало целый массив A вид 99 [1:50],   B [1:20,   1:20];
  КОД  ('ПЧ_ 10', A [5]);    КОД ('ПЧ  _ 10', A);
  КОД ('ПЧ_2-10',   B [10, 10]); КОД ('ПЧ_2-10',   B)
конец
```

В данном примере неправильной является конструкция

КОД ('ПЧ_ 10', A [5]),

поскольку в качестве фактического параметра использована переменная с индексами, имеющая по описанию указание вида.

Пример:

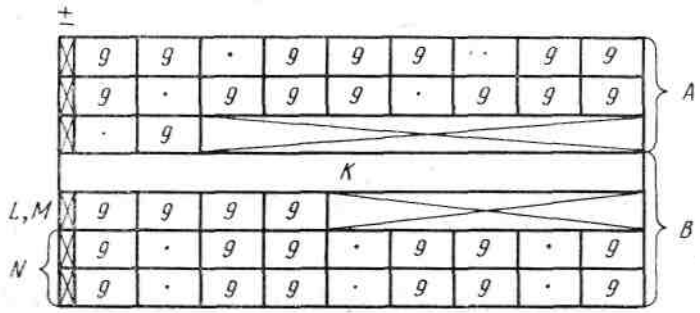
```
начало строчный A вид C  (10); строчный массив B вид C [1:8];
  ...
  КОД ('ПЧ _ АЦПУ',   A, B [5], B)
  конец
```

В данном примере все фактические параметры в операторе процедуры-кода использованы правильно.

Если в качестве фактического параметра используется групповой параметр, то его длина всегда считается равной целому числу ячеек. Например:

```
начало массив A вид 99.9 [1:5];
   составной B;
       целый K, L вид 99, M вид 99;
       массив N вид 9.9 [1:6]
       уровень;  . . .
 конец
```

Величины А и В располагаются в ячейках МОЗУ следующим образом:

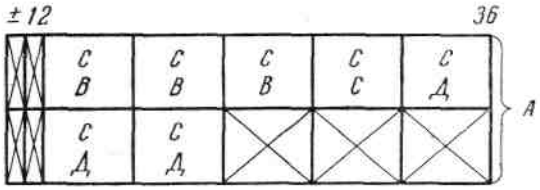


Длина массива А равна трем ячейкам (хотя в третьей ячейке остается свободное место); длина составной переменной В равна четырем ячейкам (хотя не все ячейки заполнены).

Еще пример:

```
начало составной А;
  строчный В вид С (3), С вид С, Д вид С (3) уровень;  ...
конец
```

Величина А располагается в МОЗУ следующим образом:



Длина составной переменной А равна двум ячейкам (о распределении оперативной памяти см. разд. 3,9.3).

2.17.2. Спецификации перевода '10—2' и '2—10'

Оператор процедуры-кода со спецификациями '10—2' и '2—10' может применяться к простым переменным и массивам, а также к соответствующим элементам составных переменных и составных массивов типа **целый** или **вещественный.** Его можно использовать также с составными переменными и составными массивами в качестве фактических параметров, если все их элементы имеют по описанию тип **вещественный.**

Подпрограммы перевода обеспечивают перевод чисел в соответствующую систему и размещают результаты в тех ячейках, в которых находились исходные величины. Это следует иметь в виду при дальнейшем использовании в программе величин, подвергнутых указанным преобразованиям.

Пример:

```
начало целый массив М, Я01000000 [1:100];
  КОД  ('ЧТЕНИЕ', Я01000000, М)
  КОД ('2-10', М); ...
конец
```

Предполагается, что перед выполнением оператора процедуры-кода элементы массива М были записаны в двоичном коде. В результате преобразования эти элементы, располагаясь в тех же ячейках памяти, будут записаны в двоично-десятичном коде.

2.17.3. Спецификации выдачи числовых данных на ТБПМ

Как указывалось выше это следующие спецификации:

'ПЧ' 'ПЧ _ 10' 'ПЧ _ 2-10'

По спецификации 'ПЧ' осуществляется выдача числовых данных на ТБПМ в восьмеричной системе (двоичная запись числа делится на триады, и соответствующие восьмеричные цифры печатаются на ТБПМ).

Пример:

```
начало целый А;  вещественный В, С;
   А: =25;  В: =0.5;  С: = -16.0;
   КОД ('ПЧ',  А, В, С)
конец
```

В результате работы этой программы на ТБПМ будут напечатаны следующие числа:

```
+000000000031
+777777777701
-400000000005
```

По спецификации 'ПЧ _ 10' осуществляется печать на ТБПМ чисел в десятичной системе (двоичная запись

числа делится на тетрады, и соответствующие десятичные цифры выдаются на ТБПМ).

Пример:

ключ 20 включен	*ключ 20 отключен*
+000000025	+000000025
+5000000+00	+0000000_ 5
—1600000+02	- 000000016

Предполагается, что выдаваемые на печать величины записаны в МОЗУ в двоично-десятичном коде (например, использовалась ранее спецификация '2—10', либо эти величины имеют указание вида, содержащее символ „9").

Ниже будут приведены примеры выдачи на ТБПМ чисел, являющихся значениями переменных с указаниями вида. При этом использовалась спецификация 'ПЧ _ 10':

указание вида	*число*	*выдача на ТБПМ*
$-9.9_{10}+9$	+0.181	+01_8 _ -100
	+12.95	+01 _3_ +100
	-6.0	+ -6 _0 _ +000
$+ 9.9_{10} + 9$	+357.0	+ + 3_6 _ +200
	-0.7805	+ -7_8_- 100
99.9(2)	+1200.0	+00_000000
	+ 195.3	+95 _300000
	-127.0	+27 _000000
$999.99_{10}+9$	+12.87 .	+128_70_ -1
	+489597.0	+ 489 _60 _ +3

Пример:

начало массив M **вид** —9.9_{10}+9 [1:3];
M [1]: =0.181; M [2]:=+l2.95;
M [3]: = -6.0; КОД ('ПЧ_10', M) **конец**

Результатом работы этой программы будет следующая выдача на ТБПМ:

+01 _ 8 _ - 101
+ _3_ + 1-6 _0
+ _ +0000000

Спецификация 'ПЧ 2—10' означает, что выдаваемая на печать числовая информация находится в МОЗУ в двоичной системе счисления. Производится перевод этой информации в двоично-десятичную систему и печать в десятичной системе. При этом исходная информация в МОЗУ сохраняет первоначальный вид.

Все целые и вещественные числа, использованные в записи программы на входном языке (т. е. числовые константы), в процессе трансляции переводятся в двоичную систему. Поэтому для вы. дачи их на печать в десятичной (системе (т. е. в том виде, в каком эти числа записаны в программе) следует использовать спецификацию 'ПЧ _ 2—10'. Например:

КОД ('ПЧ_ 2-10', 111111111)

На ТБПМ будет напечатано:

+ 111111111

Числа, подобные приведенному выше, удобно использовать в качестве разделителей при выдаче числовых данных на ТБПМ. Для печати пустой строки (пробела) на ТБПМ следует использовать конструкцию:

начало целый A; ...
A: =262145×262143;...
КОД ('ПЧ_ 10', A);...
конец

Пример:

начало целый A, C; **вещественный** B;
A: =25; B: = 15.2; C: =262145×262143;
КОД ('ПЧ' 30, A, B); КОД ('ПЧ_ 10', C);
КОД ('ПЧ_ 10', 30, A, B); КОД ('ПЧ_ 2-10', 555555555);
КОД ('ПЧ_2-10', 30, A, B)
конец

В результате работы этой программы на ТБПМ будут напечатаны следующие числа:

ключ 20 отключен	*ключ 20 включен*
+000000000036	+000000000036
+000000000031	+000000000031
+ 746314631404	+ 746314631404
+	+
+ 00000001_	+00000001_
+ 000000019	+000000019
+ _ 33333304	+ _ 33333304
+555555555	+555555555
+00000030	+000000030
+000000025	+000000025
+0000015_2	+1520000+02

Из приведенного примера видно, что выдачи в строках 5-й, 6-й и 7-й не имеют смысла, т. к. печатаются двоичные коды, сгруппированные в тетрады.

2.17.4. Спецификация выдачи текстовой информации на АЦПУ-128

Как было сказано выше, эта спецификация имеет вид 'ПЧ АЦПУ'.

В качестве фактических параметров в операторе процедуры-кода со спецификацией 'ПЧ _АЦПУ' могут указываться строки, переменные и массивы типа **строчный,** соответствующие элементы составных переменных и составных массивов, а также составные переменные и составные массивы, имеющие в своем составе только строчные величины, т. е. в качестве фактических параметров такого оператора могут использоваться величины, значения которых представлены в МОЗУ семиразрядными кодами АЦПУ-128. На АЦПУ нельзя выдавать значения величин типа **целый** или **вещественный**.

Для выдачи чисел на АЦПУ рекомендуется использовать оператор присваивания, обеспечивающий соответствующее преобразование типов (см. разд. 2.6.5).

Пример:

```
начало  целый  A вид 999;  строчный  B вид C (3);
  A: =375;  B:=A;
  КОД ('ПЧ _АЦПУ', B, 'КОНЕЦ 1')
конец
```

В результате работы этой программы на АЦПУ будут напечатаны следующие две открытые строки:

```
375
КОНЕЦ1
```

Каждый фактический параметр из списка фактических параметров оператора процедуры-кода со спецификацией 'ПЧ АЦПУ' печатается с новой строки. В том случае, если длина выдаваемого на АЦПУ фактического параметра составляет более 128 символов, автоматически осуществляется переход на новую строку.

Необходимо помнить, что при использовании групповых параметров в операторе процедуры-кода в соответствующей операции обмена с внешними устройствами участвует целое число ячеек (см. разд. 2.17.1). В применении к данному случаю это означает, что количество строчных символов, определяющее длину группового параметра, должно быть кратно пяти. (В том случае, если в качестве фактического параметра используется составной массив, . длина в символах каждого его элемента также должна быть кратна пяти).

Пример:

```
начало составной НАЗВАНИЕ; строчный A вид C (8),
  Б вид C (3),  B вид C (5) уровень;
  A. НАЗВАНИЕ: ='_ KOBIETA';
  Б. НАЗВАНИЕ :=' I   ';
  B.НАЗВАНИЕ :='URODA';
  КОД ('ПЧ_ АЦПУ', НАЗВАНИЕ,  A. НАЗВАНИЕ,  Б. НАЗВАНИЕ,  B. НАЗВАНИЕ)
конец
```

В результате работы этой программы на АЦПУ будут напечатаны следующие четыре строки:

```
_ KOBIETA _I_ URODA 00A5*
_ KOBIETA
_I_
URODA
```

Пример:

```
начало целый I,  L,  J вид 9 (5),  K вид 9 (7);
    составной A;
     строчный N вид C (5), ИМЯ вид C (15);
     составной массив ГР [1:5];
        строчный P вид C,  Д вид C (7)
      уровень
    уровень;
    для I: =1 шаг 1 до 5 цикл
    начало J = 1;  N. A:=J;
      ИМЯ. A:='-ОЕ УПРАВЛЕНИЕ';
      для L: = 1 шаг 1 до 5 цикл
      начало P. ГР [L].  A:='I';
        K: = I+J;  Д. ГР [L]. A:=K
        конец;
      КОД (!'ПЧ_ АЦПУ',  A,  '_')
    конец
конец ПРОГРАММЫ
```

Результатом работы этой программы будет, например, следующая таблица:

```
_ _ _ _ 1-ОЕ УПРАВЛЕНИЕ: 200:201:20A:2:H: ...
_ _ _ _  2-ОЕ УПРАВЛЕНИЕ:400:401:40A:4:H... и т. д.
```

* Четыре последних символа не определены, поскольку занимаемый ими участок оперативной памяти рабочей программой не используется (является «мертвой зоной» —см. примеры в разд. 2.17.1).

При печати появляются «лишние символы» (подчеркнуты). Их происхождение объясняется особенностями распределения памяти транслятором СТ-3 (см. разд. **2.17.1).**

2.17.5. Спецификации ввода данных с перфоленты

Спецификации ввода данных с перфоленты следующие:

'ВВОДЛ' 'ВВОДЛ _ 10-2' 'ВВОДЛ _ С

Информация, вводимая в МОЗУ с использованием спецификации 'ВВОДЛ', после ввода имеет в МОЗУ такой же вид, что и на перфоленте. Эта информация может быть отперфорирована как в десятичной, так и в восьмеричной системах.

Информация, отперфорированная в десятичной системе и введенная с использованием спецификации 'ВВОДЛ 10—2' будет представлена в МОЗУ в двоичной системе.

По спецификации 'ВВОДЛ_С информация, отперфорированная на аппарате СТА-2М, вводится в МОЗУ и перекодируется в семиразрядные коды АЦПУ-128 (см. разд. 2,17.1).

Данные, соответствующие каждому фактическому параметру, указанному в операторе процедуры-кода, должны быть отперфорированы на ленте в границах в соответствующей форме представления.

Ввод информации осуществляется через рабочее поле, рассчитанное на 2000_8 ячеек.

Числовые массивы, длина которых (с учетом упаковки) превышает длину рабочего поля, должны быть при перфорации разбиты на части ровно по 2000_8 ячеек. Эти части на перфоленте должны следовать одна за другой и перфорироваться каждая в границах. Последняя часть, очевидно, может содержать менее 2000_8 ячеек. Количество обращений к устройству ввода для каждого фактического параметра равно числу частей. Количество вводимой и пересылаемой в МОЗУ информации определяется описанием соответствующего массива. Например:

```
начало целый массив   М [1:2560];...
  КОД ('ВВОДЛ',  М);  ...
конец
```

Массив М должен быть отперфорирован на перфоленте тремя участками, каждый в своих границах. Последний участок содержит 1000_8 ячеек.

В тех случаях, когда массив вводимой с перфоленты числовой информации превышает 2000_8 ячеек, а разбивать эту информацию на части ровно по 2000_8 ячеек представляется затруднительным, рекомендуется использовать стандартную программу 157, осуществляющую ввод числовой информации с перфоленты непосредственно в МОЗУ-2, минуя рабочее поле ввода (см. разд. 3.11.2). В этом случае разбивать вводимую информацию на части не требуется.

В тех случаях, когда числовая информация вводится с перфоленты через рабочее поле ,(т. е. по одной из перечисленных в начале этого раздела спецификаций), она может быть отперфорирована как с указанием, так и без указания адресных кодов. Если информация перфорируется с указанием адресных кодов, следует иметь в виду, что каждый из участков по 2000_8 ячеек должен начинаться с адреса 00100. Адресные коды удобно попользовать в тех случаях, когда перфорируемые массивы содержат участки нулевой информации. Поскольку непосредственно перед вводом рабочее поле ввода (адрес 00100—02077) чистится, достаточно перфорировать только ненулевую информацию, рассчитав предварительно ее адреса. При работе с массивами вещественных чисел необходимо помнить, что нули, засылаемые на рабочее поле перед вводом, являются целыми нулями (код + 000000000000).

Пример:

```
начало целый А;  целый массив В [1:1300],  Е вид+99 [1:2100];

  ...

  КОД ('ВВОДЛ',  А,  В,  Е);…
конец
```

Перфолента с исходными данными для этой программы должна содержать четыре массива информации, причем каждый массив в своих границах: первый массив содержит значение А (одно число), второй и третий — значение массива В (поскольку этот массив занимает более 2000_8 ячеек), четвертый—значение массива Е (массив Е занимает менее 2000_8 ячеек, поскольку элементы его должны быть расположены по три в ячейке).

Значения А и В могут быть отперфорированы в восьмеричной или в десятичной системе. Массив Е может быть подготовлен вручную (т. е. расписан на бланках в упакованном виде), либо выдан ранее на перфоленту по спецификации 'ПРЛ' (см. 2.17.6.).

При подготовке массива, имеющего в описании указание вида, следует помнить, что в ячейке размещается 9 десятичных или 36 двоичных знаков. Знаковые разряды гари этом не используются. Знаки + и — (если это необходимо по указанию вида) следует кодировать и обрабатывать в программе специальным образом, т. к. при вводе знаковые комбинации всегда попадают в знаковый разряд и информация в ячейке смещается. Кодировка может быть, например, следующая:

знак	*код*
+	0
-	1

Тогда значения +06—11 + 17, расположенные в одной ячейке, запишутся следующим образам: +006 111017.

Перфорация первого знака «плюс десятичный» обязательна, так как в противном случае информация воспринимается при вводе как восьмеричная.

Ввод текстовой информации всегда происходит через рабочее иоле. Текстовый массив не должен превышать 2000_8 ячеек (около 7000_{10} шестиразрядных алфавитно-цифровых символов международного телеграфного кода М-2). Если вводимый массив превышает 2000_8 ячеек, происходит останов.

В качестве исходной текстовой информации допускается перфорировать все символы, имеющиеся на клавиатуре аппарата СТА-2М, за исключением круглых скобок.

Допускается использование некоторых символов, отсутствующих на клавиатуре аппарата СТА-2М. Перфорация этих символов осуществляется согласно инструкции «Запись и перфорация программ на входном языке АЛГЭМ (см. разд. 3.2).

Список дополнительных символов

‘ ’ [ ] > | ; ≠ ↑ × $_{10}$

Буква Ч при подготовке текстовой информации перфорируется как символ Х («кто там»).

Для удобства работы с текстовой информацией при ее перфорации можно использовать повторитель. Повторитель представляет собой целое десятичное число, заключенное в круглые скобки и помещенное после символа, который следует повторить. Поэтому открывающая и закрывающая круглые скобки «(» и «)» запрещены в качестве основных символов в текстовой информации.

Пример:

_ (20)	знак _	повторяется 20 раз
+ (125)	знак+	повторяется 125раз

Перед повторителем может быть указан любой символ, допустимый при перфорации текстовой информации.

Возможна многократная запись повторителя, например:

+ (10) (10) (2) - (5) (5) (1) (12)<(10) (1) (11)>(65)

Количество повторителей при такой записи не ограничено, и все они указывают число повторений одного и того же символа, стоящего перед первым повторителем. Для каждого повторителя, начиная со второго, соответствующий символ будет повторен на единицу меньше указанного в скобках числа раз. Так, в выше приведенной записи открытая строка будет содержать 20 знаков «плюс», затем 20 знаков «минус», 20 знаков «меньше» и 65 знаков «больше».

Во всех спецификациях ввода запрещено использование чисел и строк в качестве фактических параметров. Например, запись

КОД ('ВВОДЛ', 2)

не имеет смысла. Массив отперфорированной текстовой информации должен заканчиваться пятью символами «пробел».

2.17.6. Спецификации вывода данных на перфоленту

Спецификации вывода данных на перфоленту следующие:

'ПРЛ' 'ПРЛ_10' 'ПРЛ _2-10' 'ПРЛ _С'

Данные, соответствующие каждому фактическому параметру, указанному в операторе процедуры-кода, выдаются на перфоленту в границах в соответствующей форме представления.

Пример:

```
начало целый А; целый массив Д [1:1300]; целый Е вид 9;
  ...
  КОД ('ПРЛ', 29, А, Е); КОД ('ПРЛ _; 10', Д); ...
конец
```

В приведенном примере на перфоленту будут выданы четыре участка, каждый в своих границах, без адресных кодов. Выдача первых трех участков осуществляется в восьмеричной системе, а последнего — в десятичной системе (предполагается, что значения элементов массива Д находятся в МОЗУ в десятичной системе). При выдаче в восьмеричной системе каждой ячейке соответствует участок перфоленты, включающий знак, 12 восьмеричных цифр и признак «запись», а при выдаче в десятичной системе — знак, 9 десятичных цифр и признак «запись».

По спецификации 'ПРЛ_2—10' осуществляется выдача на перфоленту в десятичной системе данных, представленных в МОЗУ в двоичной системе. Перевод осуществляется в момент выдачи, данные в МОЗУ сохраняют исходный вид.

При выдаче на перфоленту значений переменных и массивов, имеющих в описании указание вида, необходимо помнить, что вывод их следует осуществлять по спецификации 'ПРЛ', т. к. при использовании спецификации 'ПРЛ 10' при последующем вводе этой перфоленты все знаки (+ или —) упакованных величин, имеющиеся в ячейке, попадают в знаковый разряд, и запись в ячейке смещается.

Пример:

```
начало целый массив Е вид +99 [1: 3]; . . .
  КОД ('ПЧ_ 10', Е); КОД ('ПРЛ _10', Е); стоп 1;
  КОД ('ВВОДЛ', Е); КОД ('ПЧ_ 10', Е); ...
конец
```

Пусть перед выдачей на печать данные в ячейке имели вид:

+ +06-11+17

Результатом работы этой программы (после ввода с перфоленты) будет:

-00006117

Этот пример следует записать так:

```
начало целый массив Е вид + 99 [1:3]; ...
  КОД ('ПЧ _ 10', Е); КОД ('ПРЛ',_10', Е); стоп 1;
  КОД ('ВВОДЛ', Е); КОД ('ПЧ_ 10', Е)
  конец
```

В результате работы этой программы на ТБПМ будут напечатаны следующие два числа:

+ +06-11 + 17
+ +06-11+17,

т. е. вывод на перфоленту был выполнен верно. При выводе на перфоленту числовых данных никаких ограничений на объем информации не накладывается. Массивы, большие, чем 2000_8 ячеек, выдаются одним участком. Поэтому при последующем вводе таких массивов следует использовать стандартную программу 157 (см. разд. 3.11.2), осуществляющую ввод массива непосредственно в МОЗУ-2, минуя рабочее поле.

По спецификации 'ПРЛ С' осуществляется вывод текстовой информации на перфоленту в коде М-2. При выводе символы АЦПУ-128, перекодированные в шестиразрядные алфавитно-цифровые символы международного телеграфного кода, засылаются на рабочее поле (2000_8 ячеек или около 7000_{10} алфавитно-цифровых знаков).

Выводимый текстовый массив не должен превышать величины рабочего поля. При превышении указанной длины происходит останов в программе 'ПРЛ С'. При выводе больших массивов следует обеспечить соответствующее число обращений к процедуре-коду со спецификацией 'ПРЛ С'.

2.17.7. Спецификации ввода данных с перфокарт

Спецификации ввода данных с перфокарт

'ВВОДК' 'ВВОДК_10' 'ВВОДК_ 10-2' 'ВВОДК_ С'

Эти спецификации обозначают соответственно следующее:'

1) ввод с перфокарт числовой информации в восьмеричной системе;
2) ввод с перфокарт числовой информации в десятичной системе;
3) ввод с перфокарт числовой информации в десятичной системе с переводом ее в двоичную систему;
4) ввод с перфокарт текстовой информации.

Числовая информация перфорируется на 80-колонных перфокартах в соответствии с инструкцией по подготовке исходных данных на перфокартах для ЭВМ «Минск-22». Ввод осуществляется по фиксированной форматной карте. На одной перфокарте всегда располагаются 8 ячеек десятичной информации или 6 ячеек восьмеричной информации. Знаки + и —, которые при вводе должны попасть в знаковый разряд ячейки, перфорируются нажатием клавиш «0» и « ≡ » соответственно. Знаки + и —, попадающие при вводе не в знаковый разряд ячейки (например, при вводе упакованной информации, имеющей указание вида), перфорируются соответственно нажатием клавиш « + » и «—».

Числовая информация вводится на рабочее поле, которое непосредственно перед вводом чистится, а затем переписывается в МОЗУ в соответствующей форме представления. Количество вводимых перфокарт рассчитывается рабочей программой, исходя из соответствующих описаний (см. также разд. 2.17.6.). Если в кармане устройства ввода перфокарт меньше, чем рассчитано по описаниям, происходит останов в стандартной программе ввода с перфокарт. Продолжение работы программы обеспечивается нажатием кнопки «пуск».

Пример:

```
начало массив М вид +9.9 [1:10];...
КОД ('ВВОДК _10', М);...
конец
```

Элементы массива М, располагаясь в пяти ячейках (т. е. на одной перфокарте), вводятся по десятичной форматной карте.

Пример:

```
начало массив А [1:2040];...
  КОД ('ВВОДК',  А);...
конец
```

В карман устройства ввода с перфокарт должно быть уложено 340 перфокарт (2040:6=340). Массив А вводится по восьмеричной форматной карте.

Спецификация 'ВВОДК _С осуществляет ввод с перфокарт текстовой информации. В качестве алфавитно-цифровых символов на перфокарты могут перфорироваться все символы, имеющиеся на клавиатуре устройства, т. е. русские и латинские буквы, цифры и специальные знаки. Латинские буквы, отличающиеся по написанию от русских, перфорируются в двух колонках: в первой — признак латинской буквы —? (знак вопроса), во второй — русская буква, соответствующая данной латинской (расположенная с ней на одной клавише). Ввод осуществляется по форматной карте. На одной перфокарте располагается 13 полных ячеек с алфавитно-цифровой информацией.

Рассчитать количество вводимых перфокарт с текстовой информацией исходя из соответствующего описания, не представляется возможным (вследствие наличия признака латинской буквы). Поэтому всегда вводится фиксированное количество (116_8 или 78_{10}) перфокарт, соответствующее длине рабочего поля ввода. Это следует учитывать при составлении программы на АЛГЭМе. Если при использовании оператора процедуры-кода со спецификацией 'ВВОДК _С число фактических параметров больше единицы, то каждый массив перфокарт следует дополнить перфокартами, заполненными целиком нулями, чтобы число перфокарт, соответствующее каждому фактическому параметру, было кратно 78. Например, для оператора

КОД ('ВВОДК _ С', А, В)

массив А следует дополнить нулевыми перфокартами до 78 перфокарт. В том случае, если после ввода массива В предполагается еще вводить информацию с перфокарт, его также следует дополнить до 78 перфокарт. Можно использовать и такую конструкцию:

```
КОД ('ВВОДК _ С',  А);   стоп 1;
КОД ('ВВОДК _С',  В);   стоп 2;...
```

В этом случае массивы А и В можно не дополнять нулевыми перфокартами, однако в момент ввода очередного массива в кармане устройства ввода должны находиться только перфокарты вводимого в данный момент массива. После ввода наступает останов (по оператору **стоп** 1**).** после которого следует положить в карман перфокарты второго массива, нажать кнопку «пуск» и т. д. Последнюю перфокарту массива, если она занята не полностью, всегда следует дополнять нулями.

2.17.8. Спецификации вывода данных на перфокарты

Спецификации вывода данных на перфокарты:

'ПРК' 'ПРК _ 10' 'ПРК _2-10' 'ПРК _С'

Эти спецификации означают соответственно следующее:

1) перфорация на перфокарты в восьмеричной системе числовых данных;
2) перфорация на перфокарты числовых данных в десятичной системе;
3) перфорация на перфокарты в десятичной системе числовых данных, представленных в МОЗУ в двоичной системе.
4) перфорация на перфокарты текстовой информации.

Количество выводимых перфокарт и вид информации (восьмеричная, десятичная или текстовая) определяются спецификацией оператора процедуры-кода и соответствующими описаниями.

При выдаче на перфокарты сначала сбрасываются две пустые перфокарты (которые при последующем вводе необходимо удалить), далее перфорируется форматная карта, а затем — информация. При перфорации осуществляется схемный контроль: при несовпадении информации, выведенной на перфокарту, с соответствующей информацией, находящейся в МОЗУ, происходит останов. Останов сигнализирует о неисправности устройства. Продолжать работу нельзя.

Вывод на перфокарты значений величин, имеющих в описании указание вида, следует осуществлять по спецификации 'ПРК _10', ввод — по спецификации 'ВВОДК 10' (с десятичной форматной картой) или 'ПРК' и 'ВВОДК' (с восьмеричной форматной картой).

2.17.9. Спецификации обмена информацией с магнитными лентами

Спецификации обращения к магнитным лентам следующие:

'ЗАПИСЬ' 'ЧТЕНИЕ'

В качестве фактических параметров в операторе процедуры-кода со спецификацией 'ЗАПИСЬ' или 'ЧТЕНИЕ' допускается использование только групповых параметров (три использовании негрупповых параметров происходит останов при трансляции).

Пример:

```
начало целый А;   целый массив В  [1:10],   Я00024000 [1:10];
  составной Я00000000;  0, 1 уровень;
  ... КОД ('ЗАПИСЬ',   А, Я00000000,   В,   Я00024000 [2]);...
конец
```

В примере допущена ошибка, так как А—простая переменная (не являющаяся групповым параметром), а Я00024000 [2] — отдельный элемент внешнего массива (также не являющийся групповым параметром).

Число пар фактических параметров в операторе процедуры-кода может быть любым. В каждой паре фактических параметров первым записывается параметр-отправитель, указывающий, откуда выбирается информация, вторым—параметр-получатель, указывающий, куда посылается информация, т. е. в операторе процедуры-кода со спецификацией 'ЗАПИСЬ' первым всегда указывается параметр, задающий внутреннюю величину, вторым—параметр, задающий внешнюю величину. В операторе процедуры-кода со спецификацией 'ЧТЕНИЕ' первым всегда указывается параметр, соответствующий внешней величине, вторым—параметр, задающий внутреннюю величину.

Количество участвующих в соответствующей операции обмена ячеек всегда определяется фактическим параметром-получателем (его описанием). Например:

```
начало массив А [1:200],   В [1:70],   Я03000000 [1:150],
  Я01000000 [1:50]; ...
  КОД ('ЗАПИСЬ',   А, Я03000000); ...
  КОД ('ЧТЕНИЕ',   Я01000000, В);   …
конец
```

В данном случае на магнитную ленту по адресу 03000000 переписывается из МОЗУ 150 первых ячеек массива А. С магнитной ленты считывается в МОЗУ массив Я01000000 полностью и еще переписывается 20 ячеек информации, находящейся непосредственно вслед за массивом Я01000000 (см. также разд. 1.5).

При записи на магнитную ленту происходит контрольное считывание со сравнением контрольных сумм записи и чтения. При несовпадении контрольных сумм происходит останов в высвечиванием на сумматоре первой команды записи. Следует повторить запись, нажав кнопку «пуск».

2.17.10. Спецификация деления целых чисел

Использование оператора процедуры-кода со спецификацией 'Д' необходимо в том случае, когда требуется найти частное и остаток от деления целых чисел. В операторе всегда указываются четыре аргумента: первый — делимое, второй—делитель, третий — частное, четвертый—остаток. Делимое и делитель могут быть целыми числами без знаков или простыми переменными типа **целый,** не имеющими указания вида. Частное и остаток могут быть простыми переменными типа **целый,** не имеющими в описании указании вида. Деление на нуль запрещено (происходит останов) .

Пример:

```
начало целый А, С, Д;
  А: =7;
  КОД ('Д', А, 2, С, Д); ...; КОД ('Д', 2, А, С, Д); ...
конец
```

В результате выполнения первого оператора процедуры-кода переменная С принимает значение, равное 3; Д равно 1. В результате выполнения второго оператора С и Д равны соответственно **0** и 2.

2.17.11. Спецификация 'ВЫХОД' (сегментация программ)

Оператор процедуры-кода со спецификацией 'ВЫХОД' имеет один фактический параметр. Этим фактическим параметром может быть либо целое без знака, либо простая переменная типа **целый,** не имеющая в описании указания вида. Например:

```
КОД ('ВЫХОД', 2)
КОД ('ВЫХОД', 4)
КОД ('ВЫХОД', К5)
```

Пусть в записи программы на АЛГЭМе оператор процедуры-кода со спецификацией 'ВЫХОД' встретился *n* раз. Тогда в процессе трансляции рабочая программа будет разбита на *n*+1 частей; при этом завершающим оператором каждой части, кроме *n+1-й,* является указанный оператор процедуры-кода. Все эти части будут последовательно записаны на магнитную ленту, начиная с адреса, заданного на наборе кода пульта управления (см. разд. 3.5 и 3.12.1). Например:

```
начало ...
  КОД ('ВЫХОД', 2);          } часть 1

  …
  КОД ('ВЫХОД', К1);         } часть 2

  …
  КОД ('ВЫХОД, 1);           } часть 3

  …
конец                        } часть 4
```

В процессе работы составленной транслятором программы действие этого оператора сводится к следующему:

Производится чтение с магнитной ленты той части программы, номер которой задается значением единственного фактического параметра данного оператора процедуры-кода;

осуществляется передача управления первому оператору вызванной части.

Следует иметь в виду, что передача управления из одной части программы в другую может осуществляться только с помощью оператора процедуры-кода со спецификацией 'ВЫХОД'.

Пример неправильной записи:

```
начало ...
    М1: ...                  } часть 1
    КОД ('ВЫХОД', 2);

    ...
    на М1;                   } часть 2
    ...
конец
```

В данном примере ошибочно использован оператор перехода (**на** М1), расположенный во второй части сегментированной программы и передающий управление метке, находящейся в первой части этой программы. Результат действия такого оператора перехода не определен.

Оператор процедуры-кода со спецификацией 'ВЫХОД' применяется в тех случаях, когда рабочая программа, составляемая транслятором, превышает 4000_8 кодов (об этом сигнализирует останов при трансляции — см. разд. 3.7).

Ниже приводится пример программы, использующей оператор процедуры-кода со спецификацией 'ВЫХОД'.

```
начало целый  I,  J,  P,  L;
    переключатель П:=М1, М2, М3;
    КОД ('ВВОДЛ_ 10-2', I);
    J: =262145×262143;  на  П[I];
    М1:КОД ('ПЧ_2-10',  111111111);
    КОД ('ПЧ_10', J); на М3;
    М2:КОД ('ПЧ_2—10',  222222222);
    КОД ('ПЧ_10',  J);
    М3:КОД ('ВЫХОД', 2);
    P:=3;  если I>2 то P:=4;
    КОД ('ВЫХОД', P);
```

```
    начало целый массив A вид 1 (3) [1:12, 1:12];
     для P: = 1 шаг 1 до 12 цикл
     для L: = l шаг 1 до 12 цикл A [P, L]: = I;
     КОД ('ПЧ', A); КОД ('ПЧ_10', J)
    конец;
  КОД ('ВЫХОД', 1);
  КОД ('ПЧ _ 2- 10', 333333333);
  КОД ('ПЧ_ 10', J);
   начало массив A вид 1 (36) [1:12];
      для P: = 1 шаг 1 до 12 цикл A [P] :=P;
      КОД ('ПЧ', A)
    конец
конец
```

Результаты работы этой программы при различных наборах значений I будут следующие:

I:1,2,3.

Результаты:

```
+111111111                    +222222222222
+                             +222222222222
+111111111111                 + 222222222222
+111111111111                 +222222222222
+111111111111                 +222222222222
+111111111111                 +222222222222
+111111111111                 +222222222222
+111111111111                 +
+111111111111                 +3333333333
+111111111111                 +
+111111111111                 +400000000001
+111111111111                  +400000000002
+111111111111                 +600000000002
+111111111111                 +400000000003
+                              +500000000003
+222222222                     +600000000003
+                              +700000000003
+ 222222222222                + 400000000004
+ 222222222222                +440000000004
+ 222222222222                .+500000000004
+ 222222222222                + 540000000004
+ 222222222222                + 600000000004
```

2. 1:3.

Результаты:

```
+333333333                    +600000000003
+                             +700000000003
+ 400000000001                +400000000004
+ 400000000002                +440000000004
+600000000002                  +500000000004
+400000000003                 +540000000004
+500000000003                 +600000000004
```

2.18. ПРИМЕРЫ ПРОГРАММ

Пример 1.

Найти минимальное число в массиве, состоящем из 10 элементов, и выдать его на ТБПМ.

```
начало целый массив A [1:10]; целый K, R;
  КОД ('ВВОДЛ_ 10-2', A); R : = A [1];
  для K: = 1 шаг 1 до 9 цикл
 начало если A [K+1]>R то на Ml;
 R:=A [K+l]; Ml: конец;
 КОД ('ПЧ_2-10', R)
конец
```

В примере используетсясписок цикла типа арифметической прогрессии.

Пример 2.

Имеется массив вещественных чисел B_1, B_2,..., B_N. Вычислить и выдать на ТБПМ сумму отрицательных чисел этого массива.

```
начало целый N;
   КОД ('ВВОДЛ _10-2', N);
   начало массив B [1 :N]; вещественный S;
       целый K; КОД ('ВВОДЛ _ 10-2', B);
       S:=0; для K: = 1 шаг 1 до N цикл
       если B [K]<0.0 то S:=S + B [K];
       КОД ('ПЧ_2-10',S)
   конец
   конец
```

Значение N, используемое в описании массива внутреннего блока, задается оператором процедуры-кода, расположенным во внешнем блоке, со спецификацией 'ВВОДЛ_10—2'.

Пример 3.

Дан массив целых чисел, состоящий из 100 элементов. Найти сумму его 50 элементов, начиная с 25-го. Сумму выдать на АЦПУ.

```
начало целый массив А [1:100];
  целый К,   СУМ1 вид +999999, СУМ;
  строчный СУМ'2 вид С  (7);
  СУМ:=0;  для К:=0   шаг 1 до 49 цикл
  СУМ: = СУМ+А [25+К];
  СУМ1:=СУМ;    СУМ2:=СУМ1;
  КОД ('ПЧ _АЦПУ', СУМ2)
конец
```

В примере использован список цикла типа арифметической прогрессии. Обратить внимание на выдачу чисел на АЦПУ.

Пример 4.

Найти в массиве, состоящем из N чисел, число, равное 2.5 с заданной точностью Е. Если число найдено, выдать его номер (номер элемента массива) на ТБПМ; если нет, выдать на ТБПМ. код 111111111.

```
начало целый N;   вещественный Е;
   КОД ('ВВОДЛ',   Е, N);
   начало массив А [1:N]; целый К;
       КОД ('ВВОДЛ_ 10-2', А);
       для К: = 1 шаг 1 до N цикл
       если abs (А [К]—2.5)≤Е то на М;
       КОД  ('ПЧ_ 2-10', 111111111); на Ml;
       М:КОД ('ПЧ_2-10', К); М1:
   конец
конец
```

Пример 5.

Выдать на АЦПУ число, месяц, год.

```
начало целый К;
  КОД ('ПЧ_АЦПУ',   '10_ОКТ._1968 г.')
конец
```

Пример 6.

В массиве, состоящем из 100 элементов, найти числа, большие 5, и выдать их массивом на перфоленту.

```
начало массив С [ I:100];    целый L;
   начало массив В [ 1: 100];    целый К;
     L:=0;    для К: = 1 шаг 1 до 100 цикл
     если С [К]>5.0 то начало L:=L+1;
     В [L] :=С [К] конец
   конец;
   начало массив В [1:L];
     КОД ('ПРЛ — 2-10', В)
   конец
конец
```

Элементы массива В, описанного во втором внутреннем блоке, совпадают с элементами массива В, описанного в первом внутреннем блоке (по распределению памяти); длина его определяется в первом блоке.

Пример 7.

Выдать на АЦПУ числовой массив, состоящий из 10 вещественных чисел, причем каждый элемент его должен печататься с новой строки.

```
начало массив А [1 : 10]; вещественный А1 вид 99.99;
целый L; строчный А2 вид С (5);
  КОД ('ВВОДЛ _ 10-2', А);
  для L: = l шаг 1 до 10 цикл
  начало А1: = А [L];   А2: = А1;
  КОД ('ПЧ _ АЦПУ', А2)
  конец
 конец
```

Пример 8.

Подсчитать значение функции У=АХ+В для Х со следующими значениями:
— 3,5; —1,0; 0; +2,0 и 2,5. Значения Х и У выдать на ТБПМ, отделяя от соседних пробелом.

```
начало вещественный X, Y, B, A;
  целый Q;   КОД ('ВВОДЛ', А, В);
  Q=262145X2×2143;
```

```
  для X: = -3.5,—1.0, 0.0, +2.0,    +2.5 цикл
  начало Y:=B + X×A;    КОД ('ПЧ _ 2- 10', X, Y);
     КОД('ПЧ _ 10', Q)
   конец
конец
```

Пример 9.

Ввести массив трехзначных целых чисел, состоящий из 1000 элементов, упаковать его и записать на магнитную ленту.

```
начало  целый  массив   A,
   B вид 999, Я00004000 вид 999 [1:1000];
   целый K;
   КОД ('ВВОДЛ_ 10-2', A);
   для K: = 1 шаг 1 до 1000 цикл
   B [K]:=A [K];   КОД ('ЗАПИСЬ', B, Я00004000)
конец
```

Пример 10.

Ввести массив трехзначных целых чисел, состоящий из 1000 элементов, упаковать его и записать на магнитную ленту.

```
начало  целый  массив A,
   B вид 999 [1 : 1000]; составной Я00004000;  0,334
   уровень; целый K;
   КОД ('ВВОДЛ _  10-2', A);
   для K: = 1 шаг 1 до 1000 цикл
   B [K]:=A [K];   КОД ('ЗАПИСЬ', B, Я00004000)
конец
```

Вводится неупакованный массив A, состоящий из 1000 элементов (каждый элемент занимает отдельную ячейку). При присваивании значений элементов массива A элементам массива B происходит их упаковка. Затем массив B записывается на магнитную ленту.

Пример 11.

Ввести массив четырехзначных целых чисел, состоящий из N элементов, упаковать его и записать на магнитную ленту.

```
начало целый N; КОД ('ВВОДЛ', N);
начало целый массив A, B вид 9999,
   Я00004000 вид 9 (4)  [1:N]; целый L;
   для L: = 1  шаг 1 до N цикл
   B [ L ] : = A [ L ];   КОД ('ЗАПИСЬ',    B, Я00004000)
   конец
конец
```

Сравнить его со следующим примером.

Пример 12.

Ввести массив четырехзначных целых чисел, состоящий из N элементов, упаковать его и записать на магнитную ленту.

```
начало целый N, K, L;
КОД ('ВВОДЛ, N):
L:=N×4;    КОД  ('Д', L, 9, K, L);
если L≠0 то K:=K+1;
начало целый массив A, B вид 9999 [l:N];
    составной Я00004000;   O, K уровень;
    КОД ('ВВОДЛ _ 10-2', A);
    для L: =1 шаг 1 до N цикл
    B [L] :=A [L];   КОД ('ЗАПИСЬ', B, Я00004000)
    конец
конец
```

При использовании конструкции

```
составной Я0000400; 0, K уровень;
```

необходим расчет K — величины внешней памяти для упакованных элементов массива A.

Пример 13.

Выдать на АЦПУ массив семизначных целых чисел, состоящий из 10 элементов, так, чтобы выдаваемые числа располагались в одну строку; (разделить соседние числа пробелами и двоеточием.

```
начало целый массив E [1:10]; целый К,
    L вид 9 (7);
    составной массив А [1:10];
    строчный В вид С (7), Д вид С (3) уровень;
    КОД ('ВВОДЛ_ 10-2', Е);
    для К: = 1 шаг 1 до 10 цикл начало
    L:=E [К];   В . А [ К ] : = L ;    Д. А [К] : ='_ | _' конец;
    КОД ('ПЧ _ АЦПУ', А)
конец
```

Обратить внимание на формирование массива исходных чисел для выдачи его на АЦПУ в одну строку.

Пример 14.

Провести расчет по одной из следующих формул в зависимости от вводимых К и Z. К принимает значения 1, 2, 3.

$$Y=F_K(Z)$$

$F_1(Z)=2,5Z$
$F_2(Z)=3, 7Z$
$F_3(Z)=4,6Z + Z^2$

Значения К и Z вводятся 5 раз.

```
начало вещественный Y, Z;  целый К, L;
   переключатель F:=F1, F2, F3;
   для L: = 1 шаг 1 до 5 цикл начало
   КОД ('ВВОДЛ _ 10-2', К, Z); на F [К];
   F1:Y:=2.5×Z; на М;
   F2:Y:=3.7×Y; на М;
   F3:Y: = 4.6×Z + Z↑2.0; М:
   конец
конец
```

В примере использован переключатель.

Пример 15.

Программа, выполняющая упорядочение (по возрастанию) значений массива А длины N с использованием одной рабочей ячейки R.

```
начало целый I, J, N; вещественный R;
   КОД ('ВВОДЛ_10—2', N);
   начало массив А [1:N];
   КОД ('ВВОДЛ_ 10-2', А);
   для I: = 1 шаг 1 до N цикл
   для J: = I+1 шаг 1 до N цикл
  начало  если  A [J]≥A [I] то на Ml;
        R:=A[I]; A[I]:=A[J]; A[J]:=R;
      Ml:  конец;
     КОД ('ПЧ_2-10', А)
  конец
конец
```

Пример 16.

Программа, выполняющая печать на АЦПУ-128 таблицы значений элементарных функций: синуса, косинуса и логарифма по натуральному основанию.

```
начало целый I, ХГ вид 99; вещественный X, ХФ вид —9.9 (5);
    строчный М вид С (75);
    составной Т;   строчный П1 вид С (7), ХГ
      вид С (2), П2 вид С (11),
      ХР вид С (8), П3 вид С (7), XS вид С (8), П4 вид С (4),
      ХС вид С (8), П5 вид С (5), XL вид С (8), П6 вид С (7)
    уровень;
    КОД   ('ПЧ _ АЦПУ', ' _ _ _ ТАБЛИЦА _ ЗНАЧЕНИЙ _ ЭЛЕМЕНТАРНЫХ _ ФУНКЦИЙ _ ', '_', '_');
    М:='-(75)';
   КОД ('ПЧ _ АЦПУ', М);
   КОД ('ПЧ _ АЦПУ',  '|_X_В_ ГРАДУСАХ _ _ |_ _X_ В_ РАДИАНАХ _ _ | _ _ _SIN_ X_ _ _ | _ _ _ COS _X_ _ _ | _ _ _ _
   LN_X_ _ _ _ _ _ _ _ |');
    КОД ('ПЧ _ АЦПУ', М);
    П1.Т: = '  |_ _ _ _ _ _  ';  П2.Т:='_ _ _ _ _ _ _ _  | _ _ _ '
    П3.Т:' _ _ _ _ _ |_ '; П4.Т: = '_ _ | _';
    П5. Т:='_ _ | _ _ '; П6.Т:='_ _ _ _ _ _ | ';
    XL.Т: = '-БЕСКОН.';
    для I: =0 шаг 3 до 90 цикл
    начало ХГ:=I;   ХГ.Т:=ХГ;
```

```
    X:=I;  X:=X×3.14159/180.0;
    XФ:=X;    XP.T:=XФ;    XФ:=sin (X);
    XS.T:=XФ;    XФ:=cos(X);    XC.T:=XФ;
    если I=0 то на Ml;
    XФ:=LN (X); XL. T:=XФ;
    Ml:КОД ('ПЧ _ АЦПУ', T)
  конец;   КОД ('ПЧ _ АЦПУ', M)
конец
```

Результатом работы этой программы является таблица элементарных функций, напечатанная на АЦПУ-128

ТАБЛИЦА ЗНАЧЕНИЙ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ

X В градусах	X В радианах	SIN X	COS X	LN X
0	0.00000	0.00000	1.00000	—БЕСКОН.
3	0.05236	0.05233	0.99863	—2.94961
6	0.10472	0.10453	0.99452	—2.25647
9	0.15708	0.15643	0.98769	—1.85100
12	0.20944	0.20791	0.97815	—1.56332
15	0.26180	0.25882	0.96592	—1.34018
18	0.31416	0.30902	0.95106	—1.15786
21	0.36652	0.35837	0.93358	—1.00370
24	0.41888	0.40674	0.91354	—0.87010
27	0.47124	0.45399	0.89101	—0.75239
30	0.52360	0.50000	0.86602	—0.64703
33	0.57596	0.54464	0.83867	—0.55172
36	0.62832	0.58778	0.80902	—0.46471
39	0.68088	0.62932	0.77715	—0.38467
42	0.73304	0.66913	0.74314	—0.31056
45	0.78540	0.70711	0.70711	—0.24156
48	0.83776	0.74314	0.66913	—0.17703
51	0.89012	0.77714	0.62932	—0.11640
54	0.94248	0.80902	0.58778	—0.05924
57	0.99484	0.83867	0.54404	—0.00518
60	1.04720	0.86602	0.50000	0.04612
63	1.09956	0.89101	0.45399	0.09491
66	1.15192	0.91354	0.40674	0.14143
69	1.20428	0.93358	0.35837	0.18588
72	1.25663	0.95106	0.30902	0.22844
75	1.30899	0.95592	0.25882	0.26926
78	1.36135	0.97815	0.20791	0.30848
81	1.41371	0.98769	0.15643	0.34622
84	1.46607	0.99452	0.10453	0.38259
87	1.51843	0.99863	0.05234	0.41768
90	1.57079	1.00000	0.00000	0.45158

Пример 17.

Программа осуществляет ввод массива целых чисел. Для каждого числа последовательно выделяются его цифры. Эти цифры печатаются. Подсчитывается и (печатается сумма цифр числа и само число.

```
начало целый N;
КОД ('ВВОДЛ_ 10—2', N);
  начало целый массив A [1:N);
    целый I, X, K, C, S, У;
    КОД ('ВВОДЛ_ 10-2', A);
    для I: = 1 шаг I до N цикл
    начало S:=0;    K: = 1;    X:=A [I];
      M:K: = K×10; КОД ('Д', X, K, C, У);
      если C≠0 то на M;
      L:КОД ('Д', K, 10, K, У);    КОД ('Д', X, K, C, У);
      X:=У;    S:=S+C;    КОД  ('ПЧ_ 2-10', C);
      если K>5 то на L;
      КОД ('ПЧ_2-10', S, A [I])
    конец
  конец
  конец
```

Пусть исходные данные имеют следующий вид:

```
   N:10
A:8, 345, 947, 5565, 10020300, 75034000', 0388, 346, 570, 231.
```

Тогда результатом работы программы будут следующие последовательности чисел:

+000000008	+000000012
+000000008	+000000345
+000000008	+000000009

+000000003	+000000004
+000000004	+000000007
+000000005	+000000020
+000000947	+000000000.
+000000005	+000000000
+000000005	+000000019
+000000005	+075034000
+000000005	+000000003
+000000020	+000000008
+000005555	+000000008
+000000001	+000000019
+000000000	+000000388
+000000000	+000000003
+000000002	+000000004
+000000000	+000000006
+000000003	+000000013
+000000000	+000000346
+000000000	+000000005
+000000006	+000000007
+010020300	+000000000
+000000007	+000000012
+000000005	+000000570
+000000000	+000000002
+000000003	+000000003
+000000004	+000000001
+000000000	+000000006
	+000000231

Пример 18.

Программа осуществляет печать на АЦПУ некоторой таблицы чисел.

```
начало целый I, J, C вид 9 (4);
    строчный массив A вид C [1:0], B вид C  (5  [1:25];
    для J: = 1 шаг 1 до 25 цикл
    начало для I: = 1 шаг 1 до 25 цикл
        начало C: =I+J;     B[I]: =C конец;
        для I: =5 шаг 5 до 120 цикл
        A [I]: = '|';
        A  [125]:' = '_ ;
        КОД ('ПЧ_АЦПУ', B)
    конец
конец
```

Результатом работы этой программы является таблица чисел (см. стр. 58-59). Следует обратить внимание на использование фиктивного массива A, начальный адрес которого в МОЗУ совпадает с начальным адресом массива B (см. разд. 3.9.3).

2:	3:	4:	5:	6:	7:	8:	9;	10:	11:	12:	13:
3:	4:	5:	6:	7:	8:	9:	10:	11:	12:	13:	14:
4:	5:	6:	7:	8:	9:	10:	11:	12:	13:	14:	15:
5:	6:	7:	8:	9:	10:	11:	12:	13:	14:	15:	16:
6:	7:	8:	9:	10:	11:	12:	13:	14:	15:	16:	17:
7:	8:	9:	10:	11:	12:	13:	14:	15:	16:	17:	18:
8:	9:	10:	11:	12	13:	14:	15:	16:	17:	18:	19:
9:	10:	11:	12:	13:	14:	15:	16:	17:	18:	19:	20:
10:	11:	12:	13:	14:	15:	16:	17:	18:	19:	20:	21:
11:	12:	13:	14:	15:	16:	17:	18:	19:	20:	21:	22:
12:	13:	14:	15:	16:	17:	18:	19:	20:	21:	22:	23:
13:	14:	15:	16:	17:	18:	19:	20:	21:	22:	23:	24:
14:	15:	16:	17:	18:	19:	20:	21:	22:	23:	24:	25:
15:	16:	17:	18:	19:	23:	21:	22:	23:	24:	25:	26:
16:	17:	18:	19:	20:	21:	22:	23:	24:	25:	26:	27:
17:	18:	19:	20:	21:	22:	23:	24:	25:	26:	27:	28:
18:	19:	20:	21:	22:	23:	24:	25:	26:	27:	28:	29:
19:	20:	21:	22:	23:	24:	25:	26:	27:	28:	29:	30:
20:	21:	22:	23:	24:	25:	26:	27:	28:	29:	30:	31:
21:	22:	23:	24:	25:	26:	27:	28:	29:	30:	31:	32:
22:	23:	24:	25:	26:	27:	28:	29:	30:	31:	32:	33:
23:	24:	25:	26:	27:	28:	29:	30:	31:	32:	33:	34:
24:	25:	26:	27:	28:	29:	30:	31:	32:	33:	34:	35:
25:	26:	27:	28:	29:	30:	31:	32:	33:	34:	35:	36:
26:	27:	28:	29:	30:	31:	32:	33:	34:	35:	36:	37:

14:	15:	16:	17:	18:	19:	20:	21:	22:	23:	24:	25:	26:
15:	16:	17:	18:	19:	20:	21:	22:	23:	24:	25:	26:	27:
16:	17:	18	19:	20:	21:	22:	23:	24:	25:	26:	27:	28:
17:	18:	19:	20:	21:	22:	23:	24:	25:	26:	27:	28:	29:
18:	19:	20:	21:	22:	23:	24:	25:	26:	27:	28:	29:	30:
19:	20:	21:	22:	23:	24:	25:	26:	27:	28:	29:	30:	31:
20:	21:	22:	23:	24:	25:	26:	27:	28:	29:	30:	31:	32:
21:	22:	23:	24:	25:	26:	27:	28:	29:	30:	31:	32:	33:
22:	23:	24:	25:	26:	27:	28:	29:	30:	31:	32:	33:	34:
23:	24:	25:	26:	27:	28:	29:	30:	31:	32:	33:	34:	35:
4:	25:	26:	27:	28:	29:	30:	31:	32:	33:	34:	35:	36:
25:	26:	27:	28:	29:	30:	31:	32:	33:	34:	35:	36:	37:
26:	27:	28:	29:	30:	31:	32:	33:	34:	35:	36:	37:	38:
27:	28:	29:	30:	31:	32:	33:	34:	35:	36:	37:	38:	39:
28:	29:	30:	31:	32:	33:	34:	35:	36:	37:	38:	39:	40:
29:	30:	31:	32:	33:	34:	35:	33:	37:	38:	39:	40:	41:
30:	31:	32:	33:	34:	35:	36:	37:	38:	39:	40:	41:	42:
31:	32:	33:	34:	35:	36:	37:	38:	39:	40:	41:	42:	43:
32:	33:	34:	35:	36:	37:	38:	39:	40:	41:	42:	43:	44:
33:	34:	35:	36:	37:	38:	39:	40:	41:	42:	43:	44:	45:
34:	35:	36:	37:	38:	39:	40:	41:	42:	43:	44:	45:	46:
35:	36:	37:	38:	39:	40:	41:	42:	'43:	44:	45:	46:	47:
36:	37:	38:	39:	40:	41:	42:	43:	44:	45:	46:	47:	48:
37:	38:	39:	40:	41:	42:	43:	44:	45:	46:	47:	48:	49:
38:	39:	40:	41:	42:	43:	44:	45:	46:	47:	48:	49:	50:

ГЛАВА 3.
РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ ТРАНСЛЯТОРА СТ-3

3.1. ПЕРФОРАЦИЯ СИМВОЛОВ ВХОДНОГО ЯЗЫКА ТРАНСЛЯТОРА

Программа, записанная на входном языке, перфорируется на аппарате СТА-2М. Ниже приводится правила перфорации всех основных символов входного языка. Символы сгруппированы по способу перфорации.

3.1.1. Перфорация букв и цифр

а) Среди букв входного языка различают следующие латинские буквы, несовпадающие по написанию с буквами русского алфавита:

D, F, G, I, J, L, N, Q, R, S, U, V, W, Z

Перечисленные буквы перфорируются следующим образом: перед каждой из них перфорируется символ «?» (знак вопроса), затем на латинском регистре перфорируется сама латинская буква.

Отметим, что из букв входного языка исключена латинская буква Y. Эта буква изображается в виде русской буквы У.

б) Русские буквы

А, Б, В, Г, Д, Е, Ж, З, И, И, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ш, Щ, Ь, Ы, Э, Ю, Я

в) Вместо буквы Ч перфорируется символ $\overset{\nabla}{\underset{\Delta}{}}$ («кто там»),

г) Цифры перфорируются обычным образом.

3.1.2. Перфорация одиночных символов

а) Следующие символы входного языка имеются на клавиатуре аппарата и перфорируются обычным образом:

$$+ - . \ = , \ ( ) / : \ \_ \ <$$

Символы пробел _ и меньше < (возврат каретки) можно перфорировать на любом регистре.

б) Символы входного языка, отсутствующие на клавиатуре аппарата СТА-2М, перфорируются в виде комбинаций существующих символов, причем такие комбинации заключаются в апострофы. В табл. 1 приводятся символы входного языка и соответствующие им комбинации символов в апострофах. Если во второй графе приводится не одна комбинация, то можно перфорировать любую из них.

Таблица 1

Символ входного языка	Перфорация символов в апострофах ''	Пояснения
1	2	3
10	'Ю' или '10'	Разделитель $_{10}$ для изображения порядка
$\times$	'.'	Знак умножения
$\uparrow$	'/'	Знак возведения в степень
[	'('	Открывающая квадратная скобка
]	')'	Закрывающая квадратная скобка
'	'((' или '1'	Открывающая кавычка
'	'))' или '2'	Закрывающая кавычка
|	':'	Вертикальный разделитель для печати на АЦПУ
:=	':=' или '=' или '=:'	Знак присваивания
;	'.,' или ',' или ',.'	Точка с запятой
$\leqslant$	'<' или '<=' или '=<' или '/<'	Знак «меньше или равно»
$\geqslant$	')/' или ')=' или '=)'	Знак «больше или равно»
$>$	')—' или '—)'	Знак «больше»
$\neq$	'=/' или '/=' или '=/='	Знак «не равно»

3.1.3. Перфорация слов-символов

При записи программы на входном языке нее слова-символы должны быть подчеркнуты. При перфорации слова-символы заключаются в апострофы. Например:

Запись на входном языке	Перфорация и соответствующие символы на контрольной ленте
начало	'НА $\overset{\nabla}{\underset{\Delta}{}}$ АЛО'
конец	'КОНЕЦ'
arctg	'A?RCT?G'

В последнем примере буквы *С* и Т перфорируются на русском регистре. Эти буквы можно перфорировать и без перехода на русский регистр, т. е. на латинском регистре, но тогда на контрольной ленте вместо буквы С

отпечатается буква S.

Как при записи, так и при перфорации слова-символы можно давать в сокращении. При сокращении слова-символа отбрасываются последние буквы. Недопустимо сокращение с использованием тире: **вещ—ный.** Минимальное количество оставляемых первых букв приведено в табл. 2. Сокращенное слово-символ при записи также подчеркивается, а при перфорации заключается в апострофы.

Таблица 2

Слово-символ (полная запись)	Перфорация первых букв сокращенного слова-символа	Минимальная запись сокращенного слова-символа
	'A'	а
abs	'A? R'	ar
arctg	'B'	в
вещественный	'ВИ'	ви
вид	'Д'	д
до	'ДЛ'	дл
для	'Е'	е
если	'ЕХ'	ех
exp	'E? N'	en
entier	'З'	з
знак	'К'	к
ко	'КОН'	кон
конец	'М'	м
массив	'НА'	нач
начало	'Н'	н
на	'П'	п
переключатель	'ПР'	пр
примечание	'С'	с
составной	'СТР'	стр
строчный	'СТ'	ст
стоп	'CO? S'	cos
cos	'Т'	т
то	'У'	у
уровень	'Ц'	ц
целый	'ЦИ'	ци
цикл	'Ш'	ш
шаг	'? S'	s
sin	'?S ?Q'	sq
sqrt	'?L'	l
ln		

Можно придерживаться также простого правила: если слово-символ состоит более чем из трех букв, то при записи или при перфорации можно сокращать это слово до трех первых букв.

3.2. ЗАПИСЬ И ПЕРФОРАЦИЯ ПРОГРАММ НА ВХОДНОМ ЯЗЫКЕ

3.2.1. Бланк

Программа или изменения к ней записываются на бланках. Можно использовать любую из двух приведенных форм бланка. Программа записывается строками.

Номером строки программы назовем последовательность символов, состоящую из символа начало строки „ ≡ " (перевод строки перфорируется на любом регистре), последовательности цифр (<целое>), дающих порядковый номер строки программы и круглой закрывающей скобки„)". Например:

≡ 139)

≡005)

≡5)

≡)*

Строкой программы назовем последовательность символов входного языка от номера до номера. Каждая строка программы имеет номер. Строка программы может быть произвольной длины и может записываться на нескольких строках бланка. Например:

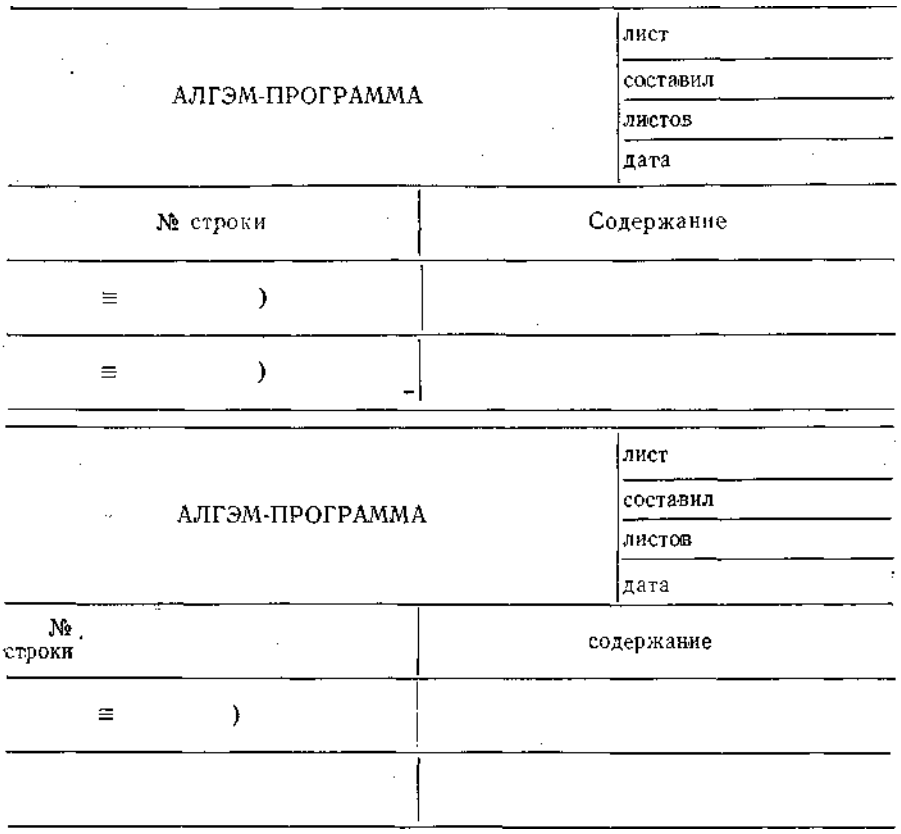
```
≡ 0) начало целый А, В;
          вещественный С, Д;
≡5) массив Е [1:100]
```

Строки бланка, на которых продолжается запись строки программы, не должны иметь номеров, т. е. символов „ ≡ " и „)".

При перфорации все символы, которые записаны на строке бланка, в том числе и номер строки программы (если он есть на строке бланка), перфорируются.

Ввод исходной программы и ее кодировку выполняет блок ввода и кодировки (БВК) транслятора.

* Последовательность цифр может быть пустой, т. е. не содержащей ни одной цифры.



Для пояснения назначения номеров строк программы, достаточно сказать следующее: номера строк программы играют ту же самую роль, что я адреса ячеек при перфорации и вводе цифровой информации.

Последовательность символов от номера до номера—это как -бы ячейка произвольной длины. БВК реализует три способа ввода и чтения строк программы:

а) При адресном (номерном) способе перед каждой строкой программы перфорируется ее номер, т. е. последовательность символов: ≡ <целое>). Это означает, что строки программы (вместе со своими номерами) могут быть отперфорированы вперемежку, т. е. не обязательно в порядке возрастания номеров строк.

Например:

```
≡ 29) A:=B×C +
≡15) начало целый A,
≡ 30) Д×E;
≡10) массив AB [1:K]
≡ 16)  B, C, Д, E;
```

б) При групповом способе последовательность символов ≡<целое>) перфорируется только для первой строки программы. Для остальных же строк программы перфорируются только два символа: „ ≡ ") (без последовательности цифр, образующих порядковый номер строки программы).

БВК присвоит таким строкам программы номера в порядке возрастания, начиная с заданного номера первой строки программы. Такое присваивание номеров будет выполняться до тех пор, пока не встретится номер строки программы, заданный в виде ≡ <целое>). Например:

```
≡ 50) A+B; A:=
≡    ) C×Д; начало
≡    ) целый A
≡60);  вещественный Д,
≡   ) M;   . . . конец
```

в) Адресно-групповой (номерно-групповой) способ включает в себя оба предыдущих способа. Независимо от того, каким способом отперфорирована исходная программа, она всегда воспринимается (транслируется, выводится на АЦПУ и на перфоленту) в порядке возрастания номеров строк программы. При наличии нескольких строк с одним и тем же номером воспринимается только последняя с данным номером.

3.2.2. Перфорация программы. Исправление ошибок перфорации и внесение изменений в программу

Перфорация программы начинается с перфорации начальной границы, далее перфорируются строки программы со своими номерами. Незначащие нули в номерах строк можно не писать и не перфорировать.

Строки программы при перфорация можно отделять друг от друга на перфоленте свободными промежутками. Во время перфорации могут быть допущены и обнаружены (по контрольной ленте) ошибки в перфорации символов. Принцип исправления таких ошибок аналогичен исправлению ошибок при перфорации цифровой информации.

Для устранения ошибок, замеченных во время перфорации, необходимо прекратить перфорацию строки программы, в которой допущена ошибка, и повторить перфорацию этой строки сначала, но обязательно с ее полным номером, т. е. последовательностью символов ≡ <целое>).

Повторную перфорацию строк программы с их полными номерами не обязательно выполнять в тот момент,

когда обнаруживаются ошибки. Все исправления можно давать и после окончания перфорации всей программы. Не является ошибкой многократная перфорация символа „ ≡ " в номере строки.

Перфорация программы и (или) исправлений заканчивается перфорацией пяти символов „ ≡ " (все на одном регистре) и конечной границы. Рекомендуется писать эти символы в конце программы:

≡249) . . **.конец конец** ≡ ≡ ≡ ≡ ≡

При вводе исходной программы в оперативную память БВК контролирует правильность перфорации символов входного языка и при наличии ошибок выдает на АЦПУ сообщение об ошибках.

Кроме того, иногда требуется вставить или удалить какой-либо оператор (группу операторов), т. е. внести изменения в программу.

Исправление ошибок, обнаруженных БВК, или внесение изменений в программу выполняются так же, как исправление ошибок, обнаруженных вовремя перфорации.

Строки программы, в которые нужно внести, изменения или исправления, перфорируются с их номерами (без начальной границы). Перфорация таких строк заканчивается пятью символами „≡" и конечной границей. Затем перфолента с изменениями подклеивается в конец перфоленты исходной программы, конечная граница которой удаляется, после чего трансляция программы выполняется снова.

Если требуется удалить всю строку программы, то для этого достаточно в изменениях отперфорировать только ее номер ≡ <целое>) без каких-либо символов за этим номером.

Поскольку нумерация строк программы может быть произвольной, то можно резервировать (оставлять неиспользованными) некоторые номера для внесения изменений

Например, четные номера можно использовать для исходной программы, а нечетные —для необходимых изменений строк программы.

Так как форма бланка довольно проста, то при записи программы можно обойтись и без бланка.

Пример записи программы: *

```
≡          ) нач цел N; КОД ('ВВОДЛ _ 10-2', N);
≡          ) примечан ВЫПОЛНЯЕТСЯ _ ВВОД _ С _
≡            ПЕРЕВОДОМ _ В_ 2 _ С/С_ЗНАЧЕНИЯ _ N;
≡          ) нач цел масс А, В [1:N], Д [1:
≡          ) 4]; цел Е, J, К, L, НП, ВП;
≡        19) КОД ('ВВОДЛ _ 10—2', А, Д);
             L:=0; для К: = 1 шаг 1 до 3 цикл
≡          ) нач J:=0;   НП: =
≡          ) Д [К];   ВП:=Д[К+1];
≡      105) для Е:=1 шаг 1 до N
             цикл нач если НП≤А [Е] то если
≡      106) А[Е]<ВП то нач J : = J + 1;
≡          ) В[J]:=А [Е]  кон кон;
           нач сост Я01000000; L, J уров;
                цел К; КОД ('ЗАПИСЬ', В, Я01000000);
≡          ) L:=L + J;
≡      2055) для К: = 1 шаг 1 до J цик
≡          ) КОД ('ПЧ _2-10', В [К]) кон кон
≡          ) кон конец ≡ ≡ ≡ ≡ ≡
```

Программа может быть отперфорирована одним массивом, если она занимает не более 35-40 бланков. В противном случае программу следует перфорировать частями.

3.2.3. Перфорация программ частями

Если программа занимает более 40 бланков или если перфолента не умещается на одном кольце, ее следует отперфорировать частями.

Каждая часть перфорируется в своих начальной и конечной границах. Каждая часть (кроме последней) и (или) изменения к ней заканчиваются перфорацией одиннадцати символов „≡" (все на одном регистре).

Последняя часть заканчивается перфорацией пяти символов

„ ≡ **".**

Разбивать программу на массивы можно в любом удобном месте.

Нумерация строк программы может быть как сквозной по всей программе, так и начинаться для каждой части программы. Следует, однако, помнить, что перфолента с изменениями или исправлениями строк программы, относящихся к какой-либо части, должна быть подклеена в конец соответствующей части.

Кроме того, при сквозной нумерации строк программы любой номер строки в первой части должен быть меньше любого номера строки во второй части. Аналогично для второй, третьей и т. д. частей.

3.2.4. Запись и использование повторителя символов

Для удобства работы с переменными типа **строчный** и строками в программе можно использовать повторитель символа. Повторитель символа состоит из целого числа, заключенного в круглые скобки.

* Для первой строки программы запись ≡) эквивалентна записи ≡ 0**).**

Пример:

```
+  (18)        означает, что знак + повторяется   18 раз,
A  (135)       означает, что буква А повторяется 135 раз.
```

Повторитель символа имеет смысл только в строках, т. е. в последовательности символов, заключенных в кавычки ".

Пример:

```
... ; строчный А вид С (40); …
A:='+(15) × (10) --_ (13)'; ...
```

В этом примере переменной А типа **строчный** будет присвоено значение, состоящее (последовательно) из 15 знаков „ + ", 10 знаков „×", двух знаков „—" и 13 пробелов.

Еще пример:

```
...; КОД ('ПЧ_АЦПУ,   'СИМВОЛ| _  (35) +
+ -(13)',   'ПЕЧАТЬ_ ×  (148)  А  (18)---(15)');…
```

Перед повторителем может стоять любой символ (см. разд. 1.2.7). Повторитель можно использовать и в текстовой информации, которая будет вводиться в обрабатываться рабочей программой. Поэтому круглые скобки ( ) нельзя использовать в иных целях.

Повторитель распространяется только на один символ, стоящий перед ним. Однако допустима и следующая запись повторителя:

```
×(15)  (10)  (4)
В (11)  (9)  (14)  (20)  (23) и т. д.
```

Количество повторителей при такой записи неограничено, и все они относятся к одному и тому же символу, стоящему перед первым повторителем. Следует иметь в виду, что для каждого повторителя, начиная со второго, соответствующий символ будет повторен не указанное в скобках число раз, а на единицу меньше.

Так, в первом примере знак „×" будет повторен 15 + 9 + 3 = 27 раз;

во втором примере буква В будет повторена 11+ 8+13+19+ + 22=73 раза.

3.2.5. Использование символа «пробел» в программах

Пробелы в программе можно использовать в любом месте и в любом количестве. Перфорируется пробел на любом регистре. Однако пробел имеет смысл только в строках, т. е. в последовательности символов, заключенных ,в кавычки. Вне кавычек пробел игнорируется.

Таким образом, не будет ошибки, если оператор присваивания

```
АВ:= С+Д×Е
```

будет записан или отперфорирован так:

```
_ _А_ _В: = _С_ _ + _Д _ _×_ _ Е_ _ _ _ _
```

Для удобства чтения исходной программы при распечатке ее на АЦПУ все пробелы сохраняются, но те пробелы, которые игнорируются, при печати никак не изображаются (т. е. изображаются как пробелы АЦПУ).

Для изображения тех пробелов, которые не игнорируются (т. е. в строках), используется знак ¯ (надчеркивание):

```
А : =  'с ¯ т ¯ р ¯ о ¯ к ¯ а'
```

3.2.6. Ограничения

При записи программы следует иметь в виду следующие ограничения:

а) Не разрешается «разрывать» номером строки последовательность символов программы, заключенную в кавычки. Это не является большим неудобством, так как всегда можно продолжить запись такой последовательности на строках бланка, не имеющих номеров. Например:

```
≡ 10) КОД   ('ПЧ_ АЦПУ', 'ПОСЛЕДОВАТЕЛЬНОСТЬ _  СИМВОЛОВ _ЗАКЛЮЧЕННУЮ_В_ КАВЫЧКИ _ НЕЛЬЗЯ _  РАЗРЫВАТЬ
_  НОМЕРОМ';
≡ 11) 'ЗАПОМНИ_ ЭТО', 'ТРАНСЛЯТОР');...
```

б) При перфорации нельзя «разрывать» номером строки слова-символы и комбинации символов, заключенные в апострофы.

в) Нумерацию строк программы можно' начинать с нуля. Номер строки не должен быть больше 8191. Строки программы, номера которых больше 8191, удаляются программой БВК. Однако возможны случаи, когда будут удаляться строки, номера которых меньше 8191. Это означает, что вводимая перфолента с исходной программой слишком длинна. В этих случаях перфоленту следует разделить на части.

г) Значение повторителя символа не должно быть больше4095, однако сумма значений повторителей, следующих друг за другом, может быть любой. Значение повторителя символа должно быть больше нуля. В случае нарушения перечисленных ограничений ББК выдает соответствующие сообщения при распечатке исходной программы на АЦПУ.

3.3. ВЫВОД НА АЦПУ И ПЕРФОЛЕНТУ ИСХОДНОЙ ПРОГРАММЫ. КОНТРОЛЬ ПРАВИЛЬНОСТИ ПЕРФОРАЦИИ СИМВОЛОВ

3.3.1. Вывод на АЦПУ

Исходная программа при трансляции всегда выводится на АЦПУ. Программа печатается на левой стороне листа АЦПУ. Если в программе имеются строки, номера которых больше 8191, то такие строки удаляются из программы. Также удаляются строки, номера которых неверно отперфорированы (вместо цифры в номере отперфорирован другой символ). Какие именно строки при этом удаляются, можно определить после печати всей программы на АЦПУ, так как удаляемые строки не печатаются.

Если в программе имеются строки с неправильно отперфорированными (или очень большими) номерами, то после окончания распечатки исходной программы на АЦПУ трансляция прекращается. Для исправления и повторной трансляции такой программы необходимо выполнить следующее. Отперфорировать удаленные строки (они отсутствуют в программе, выведенной на АЦПУ), подклеить их в конец перфоленты исходной программы (так же, как и исправления к программе) и повторить трансляцию программы, но при *включенном ключе 040.* Включение ключа 040 означает, что все строки программы с неправильно отперфорированными (или очень большими) номерами при трансляции будут опущены, но трансляция программы будет продолжена. Поэтому *первая трансляция* программы всегда должна выполняться *при выключенном ключе 040.*

Исправления номеров строк могут выполняться непосредственно и при перфорации исходной программы, как только такие ошибки обнаруживаются. Поэтому, если при первой трансляции БВК сообщает об удалении строк, необходимо сравнить исходную программу с теистом, выведенным на АЦПУ для того, чтобы убедиться, есть ли удаленные из программы строки. Если их нет (их номера уже исправлены при перфорации программы), следует повторить трансляцию при включенном ключе 040. На перфоленту (по ключу 002) исходная программа выводится без удаляемых строк. Следует помнить, что ключ 040 используется также для печати адресов меток программы, поэтому после распечатки исходной программы на АЦПУ этот ключ необходимо переключить в требуемое положение.

Программа печатается (в порядке возрастания номеров строк) листами по 25 строк в каждом. Каждая строка печатается на АЦПУ до первой ошибки перфорации в ней. На правой стороне листа АЦПУ для строки с ошибками печатается характер ошибки и дается печать всей такой строки, но не в виде последовательности символов входного языка, а в виде последовательности отперфорированных символов (то, что находится на перфоленте). По этой печати можно проверить, есть ли еще в этой строке ошибки перфорации. При этом нужно иметь в виду, что при печати ошибки **все** пробелы изображаются символом «¯» (надчеркивание), а вместо знака вопроса «?» перед латинской буквой печатается символ * (звездочка).

При печати на АЦПУ слова-символы заключаются в звездочки:

НАЧАЛО

Кроме того, все символы, которые при перфорации заключаются в апострофы, при печати на АЦПУ отделяются от соседних символов пробелами.

Некоторые символы печатаются в виде двух символов. Например, знак ≤ напечатается в виде «< = ». Сокращенные слова-символы всегда печатаются полностью. Если в программе есть ошибки перфорации или нарушены какие-либо ограничения, то трансляция программы прекращается.

Для печати на АЦПУ даты трансляции нужно после набора и занесения в МОЗУ пусковых команд транслятора на «наборе кода», начиная со старших разрядов (знаковый и четыре младших. разряда не используются), набрать в двоично-десятичном коде цифры даты. Точка набирается кодом 1110.

Например, дата .25.01.69 набирается так:

0010	0101	1110	0000	0001	1110	0110	1001	XXXX
2	5	•	0	1	•	6	9	
								не используются

Если рабочая программа должна быть записана на МЛ, то после начала печати на АЦПУ (печать даты) следует на «наборе кода» набрать соответствующую команду для записи рабочей программы на МЛ (см. разд. 3.5.3, ключ 010).

3.3.2. Вывод исходной программы на перфоленту

Исходная программа может быть выведена на перфоленту. Программа выводится на перфоратор № 1 при включенном ключе 002 и при отсутствии ошибок перфорации в ней.

Если программа разбита на части, то на перфоленту выводятся те части программы, которые не имеют ошибок перфорации.

Строки программы на перфоленте отделяются друг от друга свободными промежутками.

После перфорации всей программы трансляция продолжается.

3.3.3. Об одной возможной ошибке в программах

В любом месте программы за символом конец можно писать текст примечаний (см. разд. 215). Однако иногда за символом **конец** забывают писать символ ,,; ", хотя далее следует оператор программы. Это ошибка, так как данный оператор воспринимается как примечание, при трансляции он будет опущен, рабочая программа будет составлена неверно. Программа БВК контролирует такие случаи и при наличии их на правой стороне листа АЦПУ печатает предупреждающий текст:

←↑НЕТ СИМВ ; ЗА СИМВ *КОНЕЦ*

Стрелки обозначают, что символ ,, ; " пропущен либо в строке программы, которая напечатана на той же строке АЦПУ, что и предупреждающий текст, либо в предыдущей строке программы. БВК не может определить,

является ли это ошибкой (чаще всего это ошибка), поэтому продолжает работу. Программист (или оператор) должен сам определить, является ли это ошибкой, и решить, нужно ли продолжать трансляцию (или вывод на перфоленту) исходной программы.

Нужно отметить следующее: если в программе есть ошибки перфорации, то описанный контроль не выполняется для этих строк, так как строки программы с ошибками перфорации не просматриваются до конца.

3.4. ЗАПИСЬ ТРАНСЛЯТОРА НА МАГНИТНУЮ ЛЕНТУ

Комплект перфолент транслятора, состоящий из семи частей, предназначен для размещения транслятора на магнитной ленте 2-го ЛПМ 0-го шкафа (≈15 зон). Очередность записи перфолент на МЛ безразлична.

Для записи транслятора следует:

а) ввести очередную перфоленту. В сумматоре должна быть контрольная сумма — 7777 7777 7777;

б) выполнить «пуск» в режиме «автомат» с адреса 00111. Произойдет запись на МЛ введенной части транслятора, а затем контрольное чтение и останов.

Останов СчАК 00123 означает, что запись выполнена верно. При этом в младших (правых) разрядах сумматора находится номер части транслятора, которая записывалась, а на регистре P_1 — контрольная сумма — 7777 7777 7777.

Останов СчАК 00111 означает, что запись выполнена неверно. При этом в старших (левых) разрядах сумматора находится номер записываемой части транслятора. Для повторной записи этой части следует выполнить «пуск» в режиме «автомат».

3.5. РАБОТА С ТРАНСЛЯТОРОМ

3.5.1. Используемые устройства

В процессе работы транслятор использует рабочую ленту на 1-м ЛПМ 0-го шкафа (12—15 зон). Ошибки перфорации, синтаксические и семантические ошибки транслируемой программы, а также текст самой программы в записи на АЛГЭМе печатаются на АЦПУ, поэтому это устройство' должно быть включено. Если предполагается вывод исходной или рабочей программы на перфоленту или запись ее на магнитную ленту (см. разд. 3.9), то необходимо включить перфоратор № 1 или поставить МЛ на соответствующий ЛПМ.

Примечание. Рабочая магнитная лента должна быть размечена и проверена тестом, т. е. содержать какую-либо информацию, иначе при обращении к ней возможны сбои.

3.5.2. Пуск транслятора

Для пуска транслятора необходимо:

а) отперфорированную на бумажной ленте транслируемую программу поставить на фотоввод, включить фотоввод;

б) включить необходимые ключи и устройства (см. разд. 3.9), занести пусковые команды транслятора

```
0 0001)—47 00    0200    4000
0 0002)—45 00    2000    0001
0 0003)—30 00    0001    0000
```

и выполнить «Пуск» с адреса 00001 в режиме «автомат». *Останов СчАК 13673* сигнализирует об окончании работы транслятора. Изготовленная (рабочая) программа находится в МОЗУ-1 и выводится на ТБПМ; если нет необходимости в выводе рабочей программы на печать, то следует включить блокировку печати. Контрольная сумма рабочей программы равна —7777 7777 7777.

3.5.3. Использование ключей

При включении перечисленных ниже ключей (включается по желанию программиста) в процессе работы транслятора будет выполнено следующее:

Ключ 001— печать на АЦПУ распределения оперативной памяти (в МОЗУ-2) для рабочей программы (см. также разд. 3.9.3).

Ключ 002 — вывод исходной программы на перфоленту (перфоратор № 1) в записи на АЛГЭМе.

Примечание. При выводе исходной программы на перфоленту возможны ошибки работы перфоратора. Для повторной перфорации программы можно выполнить следующее. После начала перфорации программы на пульте набрать «Останов по команде» 01056. После окончания перфорации программы (или ее части) и останова по команде 01056 для повторной перфорации передать управление по адресу 01105.

Ключ 004 — вывод рабочей программы на перфоленту (перфоратор № 1). Контрольная сумма выведенной на перфоленту программы должна быть —7777 7777 7777. Повторная перфорация и печать рабочей программы будет выполнена, если передать управление в ячейку 13547.

Примечание. При выводе рабочей программы на перфоленту возможны сбои перфоратора. Чтобы убедиться в правильности отперфорированной программы, можно сразу же после окончания работы транслятора, не гася МОЗУ, включить блокировку записи в МОЗУ и ввести перфоленту рабочей программы. Если контрольная сумма ввода не равна —7777 7777 7777, то, выключив блокировку записи в МОЗУ, выполнить повторную перфорацию рабочей программы, т. е. передать управление в ячейку 13701. Повторить проверку перфоленты.

Ключ 010 — запись рабочей программы на магнитную ленту. При этом необходимо после занесения пусковых команд транслятора 00001—00003 на «наборе кода» набрать команду — *4700 pql (p*— номер шкафа, *q* — номер ЛПМ, *l*— номер слова магнитной ленты, на которую будет записана рабочая программа). Эта же команда будет отпечатана в последней строке массива рабочей программы на ТБПМ. Контрольная сумма записанной на МЛ программы равна -7777 7777 7777.

В случае сегментации программы (см. разд. 217.11) включение ключа 010 и набор команды —4700 *pql* на

«наборе кода» обязательны, так как изготовлены части рабочей программы последовательно записываются на МЛ.

Каждая часть сегментированной программы печатается на ТБПМ, команда —4700 *pql* (с начальным ленточным адресом записанной части программы на МЛ) печатается в конце части программы, контрольная сумма каждой части программы (при печати, перфорации, записи на МЛ) равна —7777 7777 7777. Следует иметь в виду, что на МЛ записывается по 4000_8 кодов с адреса 04000 по 07777 включительно как для несегментированной программы, так и для каждой части сегментированной программы.

Ключ 020 настраивает рабочую программу на ввод и вывод десятичной информации типа **вещественный** в форме с порядком и мантиссой. При выключенном ключе 020 рабочая программа при трансляции настраивается на ввод и вывод десятичной информации типа **вещественный** в форме с запятой.

Ключ 040 — печать на ТБПМ таблицы адресов меток транслируемой программы. Таблица меток печатается вслед за рабочей программой. Первая строка таблицы соответствует метке, описанной первой в записи на АЛГЭМе, вторая — второй метке и т. д. В каждой, строке таблицы меток содержится следующая информация: в 12 старших (левых) разрядах обозначен номер части сегментированной программы, в 12 младших (правых) разрядах находится адрес рабочей программы, соответствующий метке.

3.6. ПЕЧАТЬ ОШИБОК ТРАНСЛИРУЕМОЙ ПРОГРАММЫ.

Во время работы транслятор выявляет и печатает на АЦПУ следующие ошибки:

1) Ошибки, допущенные при перфорации исходной программы. *Останов СчАК 01065* указывает на окончание выдачи ошибок перфорации. (Об ошибках перфорации и их исправлении см. разд. 3.2.2.)

2) Синтаксические ошибки. *Останов СчАК 00505* указывает на окончание выдачи синтаксических ошибок программы.

3) Семантические ошибки. *Останов СчАК 07637 или 02317*[*] указывает на окончание выдачи семантических ошибок программы.

3.6.1. Синтаксические ошибки

При наличии синтаксических ошибок в транслируемой программе транслятор печатает их на АЦПУ и прекращает работу. Каждая синтаксическая ошибка печатается с новой строки и заканчивается символами *** (три звездочки).

В большинстве случаев смысл синтаксических ошибок очевиден. В некоторых случаях при выдаче синтаксических ошибок граничные пары в описаниях массивов и индексные выражения в переменных печатаются в обратном порядке. Кроме того, спецификации процедуры-кода и само слово КОД могут печататься в преобразованном виде.

Приведем несколько примеров, поясняющих выдачу синтаксических ошибок.

Пример 1.

```
АЛГЭМ-ПРОГРАММА 14.03.69.
  ЧАСТЬ 1
   ЛИСТ 1
≡   1) *НАЧАЛО*   A+×B
    СИНТАКСИЧЕСКИЕ ОШИБКИ ПРОГРАММЫ:
ОШИБКА: НЕТ-ОПИСАНИИ***
ОШИБКА: A+×B***
ОШИБКА: ПРОПУЩЕН СИМВОЛ *КОНЕЦ*   ***
КОНЕЦ ОШИБОК
```

Пример 2.

```
АЛГЭМ-ПРОГРАММА 14.03.69
  ЧАСТЬ 1
   ЛИСТ 1
≡        1) *НАЧАЛО*   *ЦЕЛЫИ*   K, 51;
≡        2) *ДЛЯ* K:=2 *ШАГ* 3 *ДО* 5 *ЦИКЛ* A:=A↑ *КОНЕЦ*
    СИНТАКСИЧЕСКИЕ ОШИБКИ ПРОГРАММЫ:
ОШИБКА:  *ЦЕЛЫИ*  K, 51***
ОШИБКА:  A:=A↑***
КОНЕЦ ОШИБОК
```

В данном примере в описании типа использовано число 51, и в правой части оператора присваивания пропущен показатель степени.

Пример 3.

```
АЛГЭМ-ПРОГРАММА 14.03.69
  ЧАСТЬ 1
  ЛИСТ 1
≡    I)  *НАЧАЛО*   *ЦЕЛЫИ* K, B;
≡    2)  K: = 5;
≡    3)  *НАЧАЛО*   *СТРОЧНЫЙ*   *МАССИВ* A [2:K];
```

[*] Останов 02317 сигнализирует о выдаче на АЦПУ многократно описанных идентификаторов. После нажатия кнопки «пуск» трансляция будет продолжена. Отметим, что выявляются не все ошибки, связанные с повторным описанием идентификаторов в одном блоке (см. разд. 2.14).

```
≡    4)  *ДЛЯ*    B:=3*ШАГ*   2 *ДО*   КА    [B]:=К
≡    5)  КОД ('ПЧ-2-10', А) *КОНЕЦ*   *КОНЕЦ*
      СИНТАКСИЧЕСКИЕ ОШИБКИ ПРОГРАММЫ:
 ОШИБКА:  *СТРОЧНЫЙ*    *МАССИВ* А [2:К]***
 ОШИБКА: *ДЛЯ* В:=3   *ШАГ* 2 *ДО* КА [В]:=КОД   ('ПЧ-2-10', А)***
 КОНЕЦ ОШИБОК
```

В примере 3 допущены следующие ошибки:

в описании строчного массива А не указан его вид.

в четвертой Строке примера после символа **до** пропущен символ **цикл,** поэтому в список цикла перешли все символы до символа **конец.** Кроме того, после оператора КА|В]:=К пропущен символ точка с запятой „;".

3.6.2. Семантические ошибки

При наличии семантических ошибок в программе транслятор выдает на АЦПУ соответствующие сообщения и заканчивает работу.

Каждая семантическая ошибка печатается с новой строки с указанием (в десятичной системе счисления) номера минимального блока, охватывающего текст, содержащий ошибку. В счет номеров блоков входят все составные величины.

Появление семантических ошибок связано с неправильным использованием идентификаторов в программе.

Необходимо иметь в виду, что при выдаче семантических ошибок граничные пары в описаниях массивов и индексные выражения переменных печатаются в обратном порядке. Кроме того, спецификации процедур-кодов печатаются в преобразованном виде.

Пример 1.

Число индексов не соответствует числу граничных пар в описании массивов.

```
      АЛГЭМ-ПРОГРАММА 14.03.69
    ЧАСТЬ 1
      ЛИСТ 1
≡          1) *НАЧАЛО*    *ЦЕЛЫИ* М, N, К, В;
≡          2)  КОД ('ВВОДЛ-10-2', М, N);
≡          3) *НАЧАЛО*    *МАССИВ* РЯ [1:N, 1:M];
≡          4) РЯ [К]:=0 *КОНЕЦ*   *КОНЕЦ*
      СЕМАНТИЧЕСКИЕ ОШИБКИ:
БЛОК 001 РЯ [К]:=0
КОНЕЦ ОШИБОК
```

Пример 2.

Тип переменной, указанный в описании, не согласуется с ее использованием в выражении.

```
      АЛГЭМ-ПРОГРАММА 14.03,69
    ЧАСТЬ 1
      ЛИСТ 1
≡        1)  *НАЧАЛО*    *ЦЕЛЫИ* А;
≡        2) А:=А+2.5   *КОНЕЦ*
     СЕМАНТИЧЕСКИЕ ОШИБКИ:
БЛОК 000 А:=А+2.5
КОНЕЦ ОШИБОК
```

Пример 3.

Встречается неописанный идентификатор

```
      АЛГЭМ-ПРОГРАММА -14.06.69
    ЧАСТЬ 1
      ЛИСТ 1
≡          1) *НАЧАЛО*    *ЦЕЛЫИ* А, В, С;
≡          2) А:=В+С + Д *КОНЕЦ*
      СЕМАНТИЧЕСКИЕ ОШИБКИ:
БЛОК 000 А:=В+С + Д
КОНЕЦ ОШИБОК
```

Пример 4.

В блоке многократно описан один и тот же идентификатор. В этом случае печатается номер блока и указывается тип ошибки.

Например:

АЛГЭМ-ПРОГРАММА 14.03.69

ЧАСТЬ 1

ЛИСТ 1

≡ 1) *НАЧАЛО* *ЦЕЛЫИ* А;
≡ 2) *НАЧАЛО* *ЦЕЛЫЙ* В, К, В, А;
≡ 3) В:=К *КОНЕЦ* *КОНЕЦ*
МНОГОКРАТНО ОПИСАН ИДЕНТИФИКАТОР В-БЛОК+000000000001

3.7. СПИСОК ОСТАНОВОВ ТРАНСЛЯТОРОВ СТ-3

Останов СчАК	Причина останова	Рекомендация	Программа
1	2	3	4
О0077	Программа БВК неверно считана с МЛ	Для повторного чтения — «пуск».	Программа БВК
00115	Предварительный останов перед вводом перфоленты следующей части программы	Поставить на фотоввод перфоленту следующей части программы и выполнить «пуск».	Программа БВК
00177	Неверно введена перфолента транслируемой программы	Для повторного ввода — «пуск»	Программа БВК
00374	Несовпадение контрольных сумм при чтении с МЛ программы компилятора синтаксического контроля	»	Компилятор синтаксического контроля
О0404	Переполнение магазина необработанных определений	Укорачивать арифметические выражения строки; уменьшать количество операторов (поставить дополнительные **начало** и **конец)**	Компилятор синтаксического контроля
О0505	Конец выдачи синтаксических ошибок программы	Исправить ошибки и транслировать программу сначала	Компилятор синтаксического контроля
00717	Несовпадение контрольных сумм при чтении с МЛ входной строки анализатора	Для повторного чтения — «пуск»	Анализатор
О1065	Конец выдачи ошибок перфорации или нарушены ограничения (см. 3.2.6.)	Для повторного транслирования программы исправить ошибки	Программа БВК
01103	Переполнение магазина анализатора, в котором хранятся параметры для подцелей и номера определений выходной строки	Уменьшить длину операторов, выражений, увеличив количество операторов транслируемой программы	Анализатор
01127	Несовпадение контрольных сумм при чтении с МЛ входной строки анализатора	Для повторного чтения— «пуск»	Анализатор
01174	Исходная программа не соответствует синтаксису входного языка (грамматический анализ транслируемой программы не может быть завершен)		»
01311	Переполнение магазина анализатора, в котором хранятся параметры для подцелей и номера определений входной строки	Уменьшить длину выражений, увеличив количество операторов программы	Анализатор
01434	Несовпадение контрольных сумм при чтении с МЛ программы анализатора	Для повторного чтения — «пуск»	»
01440	Исходная программа не соответствует синтаксису входного языка (грамматический анализ транслируемой программы не может быть завершен)		»
01652	Переполнение магазина количества блоков и составных переменных транслируемой программы	Уменьшить в программе количество блоков и составных переменных. Максимальное количество их равно 179	»
01771	Переполнение поля метакомпонентов	Укорачивать идентификаторы, уменьшать количество граничных пар в описаниях массивов и составных массивов	Составитель таблиц
02153	Очень много идентификаторов, начинающихся с одной и той же буквы. (См. печать на АЦПУ)	Начинать идентификаторы с разных букв	»
02317	Многократно описан идентификатор в блоке	Исправить ошибку в программе. Номер блока и идентификатор указаны в выдаче на АЦПУ	»
02324	Переполнение списка U или V	Уменьшить количество идентификаторов в программе	»
02565	Переполнение матрицы Т по строкам	Уменьшить количество идентификаторов в программе	Составитель таблиц

Останов СчАК	Причина останова	Рекомендация	Программа
1	2	3	4
03014	Несовпадение контрольных сумм при чтении с МЛ программы анализатора и составителя таблиц	Для повторного чтения — «пуск»	Анализатор и составитель таблиц
03024	Переполнение таблицы Z	Уменьшить, количество идентификаторов в программе	Составитель таблиц.
06537	Несовпадение контрольных сумм при чтении программы компилятора и таблицы Z для первого просмотра с МЛ	Для повторного чтения —«пуск»	Компилятор
06564	Переполнение поля метакомпонентов	Укорачивать идентификаторы, уменьшать количество граничных пар в опис. массивов и составных массивов	Программа объединения
06624	Несовпадение контрольных сумм при чтении глобальных правил и программы «объединения»	Для повторного чтения — «пуск»	»
07563	Несовпадение контрольных сумм при чтении с МЛ программы выдачи семантических ошибок	»	Компилятор
07614	Несовпадение контрольных сумм при записи выходной строки компилятора при первом просмотре	Для повторной записи —• «пуск»	Компилятор
07622	Несовпадение контрольных сумм при записи выходной строки компилятора на втором просмотре	»	»
07637	Конец выдачи семантических ошибок	Исправить в транслируемой программе семантические ошибки	»
07703	Номер метакомпонента больше 32	Уменьшить количество граничных пар в описаниях массивов и составных массивов	»
07704	Номер метки больше, чем 377_8	При останове на 1-м просмотре: сократить число меток и описаний переключателя. При останове на 2-м просмотре: сократить число условных операторов и опер.цикла	»
07770	Не хватает рабочих ячеек для рабочей программы	Укорачивать арифметические выражения; сокращать количество фактических параметров в процедурах-кодах	»
07745	Переполнение матрицы Y, таблицы Z или магазина необработанных определений	Укорачивать арифметические выражения, строки, уменьшать количество операторов (поставить дополнительные **начало** и **конец;** сократить количество описаний	Компилятор
12004	Несовпадение контрольных сумм при чтении программы перевода с промежуточного языка на машинный	Для повторного чтения — «пуск»	Программа перевода с промежуточного языка на машинный
12027	Несовпадение контрольных сумм при считывании с МЛ входной строки на промежуточном языке	»	»
12056	Произошло наложение полученной транслятором программы и ее констант	Объем задачи больше допустимого 4000_8 кодов Сегментировать задачу и транслировать сначала	»
13433	Для сегментированной рабочей программы не включен ключ 010 для записи частей программы на МЛ	Включить ключ 010, проверить команду—4700 *pql* на «наборе кода» и передать управление в ячейку 13431	»
13673	Работа транслятора закончена	Решение задачи по полученной программе. Пусковой адрес 07727. Включить необходимые внешние устройства	»
13711	Попытка записать рабочую программу на МЛ транслятора	На «наборе кода» исправить команду— 4700 *pql* для записи рабочей программы на МЛ и передать управление в ячейку 13670	»
13730	В процедурах 'ЗАПИСЬ', 'ЧТЕНИЕ' используются переменные, числа или строки	Исправить ошибки в транслируемой программе и транслировать сначала	»
14017	Число символов в указании вида переменных больше девяти	Исправить указание вида переменных. Задачу транслировать сначала	

Останов СчАК	Причина останова	Рекомендация	Программа
1	2	3	4
14027	Попытка записать сегментированную рабочую программу на МЛ транслятора	На «наборе кода» исправить команду—4700 *pql* для записи рабочей программы на МЛ и передать управление в ячейку 13431	Программа перевода с промежуточного языка на машинный
14164	Неправильное использование фактических параметров в операторе процедуры-кода	Исправить оператор процедуры-кода. Задачу транслировать сначала	»
14176	Неправильное использование фактических параметров в операторе процедуры-кода со спецификацией 'Д'	»	»

3.8. ОБСЛУЖИВАЮЩИЕ ПРОГРАММЫ ТРАНСЛЯТОРА

В целях удобства работы с транслятором в его состав включены следующие обслуживающие программы: проверки сохранности транслятора на МЛ, дублирования транслятора с одной МЛ на другую, проверки перфорации рабочих программ, перфорации, печати на АЦПУ и проверки перфолент частей транслятора, включения-исключения стандартных программ в БСП транслятора.

При работе обслуживающих программ магнитная лента транслятора должна находиться на 2-м ЛПМ 0-го шкафа.

3.8.1. Тест проверки сохранности транслятора на магнитной ленте

Во время эксплуатации магнитной ленты с транслятором следует периодически проверять состояние записанной на ней информации. Проверка осуществляется с помощью теста транслятора, который обеспечивает вызов в МОЗУ поочередно всех семи частей транслятора и проверку их контрольных сумм.

Для. работы теста транслятора следует занести с пульта и выполнить в режиме «автомат» команды:

```
00011)-47    00    0200    4000
00012)-45    00    0075    0001
00013)-30    00    0011    0000
```

Останов СчАК 00031 последует при несовпадении контрольных сумм какой-либо части. В старших разрядах сумматора будет находиться номер части транслятора, для которой не совпали контрольные суммы. Для повторного чтения этой части следует выполнить «пуск».

При двукратном останове по одной и той же части транслятора проверка следующих частей будет выполнена, если произвести «пуск».

Останов СчАК 00011 означает конец работы теста транслятора (при нажатии кнопки «пуск» произойдет повторная работа теста). В младших разрядах сумматора будут находиться номера тех частей транслятора, для которых контрольные суммы совпали. Неверно считанные с МЛ части транслятора следует перезаписать.

3.8.2. Дублирование транслятора на магнитные ленты

Для дублирования транслятора на другие магнитные ленты необходимо занести с пульта и выполнить в режиме «автомат» следующие команды:

```
0000l)-47    00    pq00    3210
00002)-45    00    0035    0004
00003)-30    00    0001    p1q100,
```

где p и q — номер шкафа и номер ЛПМ, - соответственно той МЛ, с которой производится дублирование,

p_1 и q_1 — номер шкафа и номер ЛПМ, соответственно, на МЛ которых производится дублирование.

Останов СчАК 00034 означает конец дублирования транслятора.

Останов СчАК 00027 возникает при несовпадении контрольных сумм при записи-чтении частей транслятора на МЛ. Для повторения записи-чтения следует выполнить «пуск».

Примечание. Для дублирования МЛ с транслятором может находиться на любом ЛПМ, т. к. p и q задаются с пульта.

3.8.3. Проверка перфорации рабочих программ

В разделе «Использование ключей» (3.9.2) будет отмечено, что рабочая программа выводится на перфоленту при включенном ключе 004. Будет также описан способ проверки перфоленты рабочей программы. Другой способ проверки заключается в сравнении 2-х экземпляров рабочей программы, один из которых находится на МЛ (записан на МЛ во время трансляции по ключу 010), а другой выдан на перфоленту по ключу 004. Такую проверку выполняет данная программа. Для проверки перфоленты рабочей программы необходимо выполнить следующее:

а) поставить МЛ с проверяемой рабочей программой на соответствующий ЛПМ, включить перфоратор № 1 и ТБПМ;

б) занести с пульта команды:

```
00001)-47    00    0200    3100
00002)-45    00    0110    0001
00003)-30    00    0001    0000
```

в) на «наборе «кода» набрать команду —4700 *pql,* т. е. ту команду, которая набиралась при трансляции для

записи на МЛ рабочей программы;

г) поставить перфоленту с рабочей программой на фотоввод включить фотоввод и выполнить «пуск» с адреса 00001 в режиме «автомат».

Останов СчАК 00015 означает, что контрольная сумма рабочей программы, считанной с МЛ, не равна —7777 7777 7777. Для повторного считывания следует выполнить «пуск».

Если контрольная сумма программы верна, то происходит ввод перфоленты. При этом на печать выводятся контрольные суммы рабочей программы,- считанной с МЛ и введенной с перфоленты.

Останов СчАК 00032 сигнализирует о том, что рабочие программы на МЛ и перфоленте не соответствуют друг другу: поставлена не та МЛ, перфолента и т. д., либо неверно отперфорирована ячейка 07727 (содержимое этой ячейки — передача управления на начало программы). При этом на печать выводится содержимое ячейки 07737, считанной с МЛ или введенной с перфоленты.

Если после останова СчАК 00032 выполнить «пуск», то произойдет сравнение кодов программы, печать и перфорация исправлений. По количеству выведенных на печать кодов можно определить, следует ли далее выдавать исправления на перфорацию. . Если нет останова СчАК 00032, то происходит сравнение содержимого соответствующих ячеек рабочей программы. Для не совпавших ячеек на печать выводятся три строки: адрес ячейки, содержимое данной ячейки (верное), считанное с МЛ, и содержимое ячейки, введенное с перфоленты. После окончания печати последует перфорация соответствующих ячеек с адресными кодами.

Останов СчАК 00100 сигнализирует об окончании работы программы проверки. Выданную перфоленту (исправления) следует подклеить в конец перфоленты рабочей программы и выполнить проверку сначала. Следует иметь в виду, что сумма ввода перфоленты рабочей программы при этом, очевидно, изменится.

3.8.4. Программа перфорации, печати на АЦПУ и проверки перфолент транслятора

Для всех семи частей транслятора программа выполняет перфорацию, печать на АЦПУ в форме бланка в двух экземплярах, проверку перфолент (сравнение соответствующей информации на МЛ и перфоленте) с печатью на ТБПМ ошибок и перфорацией исправлений ошибок.

Для работы программы с пульта заносятся и выполняются в режиме «автомат» пусковые команды:

00150) - 47	00	0200	3250
00151) - 45	00	0525	0153
00152) - 30	00	0150	000N

где N — номер части транслятора (N=1, 2, 3, 4, 5, 6, 7).

Останов СчАК 00323 сигнализирует об окончании работы программы. На сумматоре в младших разрядах находится номер части, для которой выполнялась программа.

Программа работает в следующих режимах:

а) Вывод на перфоленту

Для вывода на перфоленту следует:
выключить ключи 001 и 002,
включить перфоратор № 1,
выполнить «пуск» с адреса 00)150 в режиме «автомат». Последует перфорация части транслятора с заданным номером N.

Для удобства внесения изменений перфолента выводится массивами по 60_8 ячеек со свободным промежутком между массивами. Перед каждым массивом имеется адрес сопровождения (адресный код), причем младшая цифра адреса сопровождения совпадает с номером перфоленты,

После перфорации следует выполнить проверку выданной перфоленты по п. *в)* Сумма ввода перфоленты должна быть —7777 7777 7777.

Примечание. Если во время перфорации произошел обрыв или не хватило перфоленты, то для продолжения перфорации выполнить следующее: остановить машину, в ячейку 00022 по второму адресу занести адрес сопровождения массива, вывод которого не закончен, и передать управление в ячейку 00300.

б) Печать бланков перфоленты на АЦПУ

Для выдачи бланков следует:
выключить ключ 002,
включить ключ 001 и АЦПУ,
выполнить «пуск» с ячейки 00150 в режиме «автомат». Произойдет печать на АЦПУ в форме бланков в двух экземплярах содержимого перфоленты с заданным номером. На каждом бланке младшая цифра первого адреса сопровождения также совпадает с номером части.

Примечание. Если во время перфорации (см. п. *а)* включить ключ 001, то по окончании перфорации последует печать на АЦПУ. Каждый массив в 60_8 ячеек на перфоленте соответствует одному бланку печати на АЦПУ,

в) Проверка перфолент транслятора

Для проверки сравниваются ячейки, считанные с МЛ и введенные с перфоленты. Проверка осуществляется следующим образом: включить ключи 001, 002, ТБПМ, перфоратор № 1 и АЦПУ,
поставить на фотоввод проверяемую часть,
включить фотоввод,
выполнить «пуск» с ячейки 00150 в режиме «автомат».

Останов СчАК 00214 указывает на несоответствие заданного номера номеру проверяемой части. На сумматоре по первому адресу находится заданный номер части, а по второму адресу — номер части на фотовводе. В этом случае необходимо задать другой номер или поставить другую часть и выполнить проверку сначала.

Для каждой части на ТБПМ печатаются: номер, контрольная сумма части, считанной с МЛ и введенной с

перфоленты. При наличии ошибок на перфоленте печатаются следующие три строки: в первой строке старшая цифра указывает номер части, четыре младшие цифры указывают адрес не совпавших ячеек. Вторая и третья строки дают верное (считанное с МЛ) и неверное (с перфоленты) значения данной ячейки соответственно. По окончании печати на ТБПМ последует перфорация и печать на АЦПУ исправлений. Причем, перфорируются и печатаются с адресами сопровождения те массивы (по 60s ячеек), в которых обнаружены ошибки. Такая выдача позволяет заменять соответствующие бланки или массивы на перфоленте (массивы с ошибками удаляются) без изменения контрольной суммы ввода перфоленты —7777 7777 7777.

Примечание 1. Если не вносились изменения на МЛ транслятора и имеется выдача бланков перфолент на АЦПУ, то включение ключа 001 и АЦПУ при проверке частей необязательно, так как в этом случае вновь выданные бланки не будут отличаться от прежних.

2. Для повторной перфорации (печати на АЦПУ) исправлений следует передать управление в ячейку 00245.

3.8.5. Программа включения-исключения стандартных программ в БСП транслятора

Для расширения возможностей транслятора в его БСП можно включать новые стандартные программы, составленные в кодах машины. Требования по составлению и оформлению новых СП изложены в разделе 3.10.2. Каждой СП присваивается восьмеричный номер N, причем $1 \le N \le 176$.

Включенным СП присвоены следующие номера: 1, 2, 15, 16, 20, 21, 22, 25÷32, 36, 60÷76, 122÷126, 130, 131, 134÷160. Новым СП можно присваивать любые номера из указанного выше интервала кроме перечисленных.

Данная программа служит для включения-исключения СП с заданным номером. Для работы программы следует занести с пульта и выполнить в режиме «автомат» пусковые команды:

```
00200) -47      00      0200      2520
00200) -45      00      0360      0203
00202) -30      00      0200      0NCП,
```

где NCП — номер включаемой-исключаемой стандартной программы.

Останов СчАК 00230 указывает на окончание работы программы. На сумматоре находится номер СП, для которой выполнялась программа.

Останов СчАК 00232 возникает в следующих случаях:

не задан номер СП; на сумматоре—нули;

номер СП>176; на сумматоре — заданный номер СП;

СП с заданным номером включать-исключать запрещено; на сумматоре — заданный номер СП.

а) Исключение СП

При исключении СП освобождается место для других СП. Чтобы исключить СП, следует:

выключить ключ 001,

включить ТБПМ и выполнить «пуск» с адреса 00200 в режиме «автомат».

На ТБПМ печатается номер СП и нулевая строка — признак исключения из оглавления БСП строки, содержащей информацию о стандартной программе с заданным номером.

б) Включение СП

Для включения СП следует:

включить ключ 001 и ТБПМ,

поставить перфоленту с СП на фотоввод, включить фотоввод,

выполнить «пуск» с адреса 00200 в режиме «автомат».

Останов СчАК 00324 возникает при несовпадении контрольных сумм при вводе перфоленты СП. Для повторного ввода перфоленты выполнить «пуск».

Останов СчАК 00374 возникает в случае несовпадения контрольных сумм при записи и контрольном чтении СП. Для повторения записи-чтения выполнить «пуск».

Останов СчАК 00550 возникает при несовпадении контрольных сумм при записи-чтении оглавления БСП. Для повторения записи-чтения выполнить «пуск».

Для включенной СП печатаются две строки. Первая — номер СП, вторая — строка из оглавления БСП. Старшие (левые) 12 разрядов этой строки дают полную длину, а остальные разряды — ленточный адрес включенной СП.

При включении СП происходят изменения в 6-й и 7-й частях МЛ транслятора. Для того чтобы иметь эти изменения на перфолентах, следует выполнить проверку 6-й и 7-й частей способом, описанным в предыдущем пункте.

Останов СчАК 00421 возникает в том случае, когда для включаемых СП не хватает места на МЛ в объеме памяти, отведенном для БСП. При выполнении «пуск» произойдет включение СП, т. е. такие СП будут работать в составе транслятора. Однако на МЛ эти СП будут находиться вне объема памяти БСП и при выдаче (или проверке) 7-й части не войдут в ее состав. Если последнее нежелательно, то рекомендуется исключить ненужные СП.

Примечание. Перфолента включаемой СП должна содержать только текст СП, т. к. после ввода полная длина СП определяется по первой занятой ячейке, начиная с конца МОЗУ-1.

3.9. ОПИСАНИЕ РАБОТЫ СОСТАВЛЕННЫХ ТРАНСЛЯТОРОМ ПРОГРАММ

В конце работы транслятора составленная им рабочая программа, расположенная в МОЗУ-1, выдается на узкую печать, а также в зависимости от положения ключей при трансляции может быть выдана на перфорацию (ключ 004 включен) или записана на магнитную ленту по адресу, набранному на «наборе кода» (ключ 010 включен). Контрольная сумма программы, выданной на перфоленту, на ТБПМ или записанной на магнитную ленту, равна —7777 7777 7777.

3.9.1. Пуск программ

Пусковой адрес программы —07727 (в ячейке 07727 записана команда передачи управления на начало программы).

Конец работы —07771.

Работа сегментированной программы начинается с вызова первой ее части с магнитной ленты. Ленточный адрес для вызова тот же, что был указан на «наборе кода» при трансляции программы (команда, которую следует занести в ячейку 00001 для вызова программы, печатается на ТБПМ после текста программы). Следовательно, для передачи управления сегментированной программе с пульта должны быть занесены и выполнены в режиме «автомат» следующие команды:

```
00001)-47    00    pql
00002)-45    00    4000    4000
00008)-30    00    0001    0000
00004)-30    00    7727    000O
```

Во время работы составленной транслятором программы должны быть включены те устройства, к которым имеются обращения (устройства ввода, ТБПМ, АЦПУ, перфораторы, НМЛ), а также ключи, используемые в программе. Рабочая программа использует библиотеку стандартных программ транслятора, записанную на МЛ транслятора. Поэтому во время работы программы МЛ с транслятором должна находиться на 2-м ЛПМ 0-го шкафа. Работа программы начинается переписью в МОЗУ-1 интерпретирующей системы (ПС), через которую осуществляется вызов необходимых стандартных программ (СП).

Рабочее поле ИС располагается с адреса 02457 до начала рабочей программы. Рабочая программа и ее константы размещаются в конце МОЗУ-1. Максимальная длина программы вместе с константами (или одной части ее при сегментации) составляет 4000_8 ячеек. (Следовательно, минимальная длина рабочего поля — 1321_8 ячейка).

Вызов СП происходит в порядке их использования в готовой программе. В случае сегментированной программы при вызове очередной части вновь вызывается ИС и чистится таблица информации ранее вызванных СП.

Выход из ИС в рабочую программу осуществляется по ячейке 00016.

3.9.2. Использование ключей

При работе составленной транслятором программы в операторах перехода по ключу могут быть использованы любые ключи (см. разд. 2.10), кроме ключей 100 и 020, действие которых точно определено:

ключ 100. Последними командами рабочей программы всегда являются команды перехода по ключу 100 и останова. При отключенном ключе 100 работа заканчивается командой останова, при включенном— управление передается в ячейку 10400 (МОЗУ-2), где может располагаться любая, составленная в кодах ЭВМ программа;

ключ 020. Определяет способ ввода-вывода вещественных чисел в рабочей программе. Если ключ 020 не включен, то ввод и вывод вещественных чисел производятся в системе «с запятой». Если ключ 020 включен, то ввод и вывод вещественных чисел производятся в системе с порядком и мантиссой. Ключ 020 в зависимости от способа ввода-вывода должен быть включен или выключен также при трансляции рабочей программы.

3.9.3. Распределение оперативной памяти в процессе работы программы

Простые переменные типа **целый** и **вещественный,** независимо от наличия указания вида, занимают полные ячейки и располагаются в оперативной памяти, начиная с ячейки 17777, в сторону уменьшения номеров ячеек.

Простые переменные типа **строчный,** а также массивы, составные переменные и составные массивы, идентификаторы которых не начинаются с буквы Я, располагаются в оперативной памяти, начиная с ячейки 10400 в сторону увеличения номеров ячеек.

Таблица распределения оперативной памяти (выдается на АЦПУ по желанию программиста, см. разд. 3.5.3) содержит строки двух видов:

а) для простых переменных типа **целый** и **вещественный** выдаются номера ячеек оперативной памяти, отведенных для значений соответствующих переменных;

б) для простых переменных типа **строчный,** массивов, составных переменных и составных массивов выдаются номера ячеек, в которых в процессе работы программы хранятся динамические начальные адреса соответствующих величин, причем для простых переменных типа **строчный** указанная ячейка содержит номер разряда относительно начала МОЗУ-2, начиная с которого помещается соответствующая переменная.

Ячейки 00001÷00077 и 10000÷10377 используются в качестве индексных и рабочих.

Ячейки 00100÷02077 используются как рабочее поле ввода-вывода информации (см. разд. 2.17.5—12.17.8).

Ячейки 02100÷02456 заняты интерпретирующей программой, организующей работу библиотеки стандартных программ.

Ячейки с 02457 до начала рабочей программы отведены под рабочее поле интерпретирующей программы. (см. разд. 3.9.1).

3.9.4. Остановы при работе составленных транслятором программ

При работе составленной транслятором программы могут быть следующие остановы:

Останов в программе с высвечиванием на СМ кода + 777700000000 происходит в случае неверно считанной ИС. Следует повторно передать управление по адресу 07727.

Останов СчАК 02237 указывает на то, что СП неверно считана с МЛ. Номер ее высвечивается на СМ. Следует выполнить «пуск» (для повторного чтения СП).

Останов СчАК 02254 означает, что СП, номер которой высвечивается на СМ, не умещается на рабочем поле ИС. Следует уменьшить длину рабочей программы (например, сегментировать ее).

Останов в рабочей программе с высвечиванием на СМ кода +00000000 7777 наступает в тех случаях, когда для переменных и массивов, описанных в программе, необходима большая оперативная память, чем та, которая отведена в МОЗУ-2 (адреса 10400÷17777 —см. разд. 3.9.3).

Останов в программе с высвечиванием на СМ кода —4700 *pql* происходит в случаях неверно считанной или

записанной на магнитную ленту информации. Следует выполнить «пуск» (для повторного чтения или повторной записи).

3.10. БИБЛИОТЕКА СТАНДАРТНЫХ ПРОГРАММ

№ СП	Наименование СП
1	10 → *2* перевод числа из 10 с/с в 2 с/с с фиксированной запятой
2	2 → 10 перевод числа из 2 с/с в 10 с/с с фиксированной запятой
15	10 → *2* перевод целого числа из 10 с/с в 2 с/с
16	2 → 10 перевод целого числа из 2 с/с в 10 с/с
20	10 → *2* перевод числа из 10 с/с в 2 с/с с плавающей запятой
21	2 → 10 перевод числа из 2 с/с в 10 с/с с плавающей запятой
22	текстовой ввод с перфоленты (в тексте не допускаются повторители символов).
25	$\sqrt{x}$
26	E^x
27	*In x*
30	*sin x*
32	*cos x*
36	*arctg x*
60	10 → 2 перевод числа «с запятой» из 10 с/с в 2 с/с
61	10 → 2 перевод группы чисел «с запятой» из 10 с/с в 2 с/с
62	2 → 10 перевод группы чисел «с запятой» из 2 с/с в 10 с/с
63	10 → *2* перевод группы чисел из 10 с/с в 2 с/с с плавающей запятой
64	2 → 10 перевод группы чисел из 2 с/с в 10 с/с с плавающей запятой
65	10 → 2 перевод группы целых чисел из 10 с/с в 2 с/с
66	2 → 10 перевод группы целых чисел из 2 с/с в 10 с/с
67	чтение информации, контрольная сумма которой дополнена до кода— 7777 7777 7777 с магнитной ленты
70	10 → 2 перевод группы чисел «с запятой» из 10 с/с в 2 с/с (количество чисел задано по адресу)
71	запись информации на МЛ с дополнением ее контрольной суммы до кода —7777 7777 7777
72	подвод зоны НМЛ
73	2 → 10 перевод группы целых чисел из 2 с/с в 10 с/с (количество чисел задано по адресу)
74	10 → 2 перевод группы целых чисел из 10 с/с в 2 с/с (количество чисел задано по адресу)
75	2 → 10 перевод группы чисел «с запятой» из 2 с/с в 10 с/с (количество чисел задано по адресу)
76	2 → 10 перевод числа «с запятой» из 2 с/с в 10 с/с
122	текстовой ввод с перфоленты
123	текстовой вывод на перфоленту
124	пересыл строчных величин
125	возведение в степень целых чисел
126	числовой ввод с перфоленты-
130	2 → 10 перевод группы чисел из 2 с/с в 10 с/с с плавающей запятой (количество чисел задано по адресу)
131	10 → 2 перевод группы чисел из 10 с/с в 2 с/с с плавающей запятой (количество чисел задано по адресу)
134	преобразование числовых значений в строчные
135	преобразование строчных значений в числовые
136.	преобразование вещественных значений в целые
137	преобразование целых значений в вещественные
140	деление целых чисел с получением частного и остатка
141	печать на АЦПУ
142	преобразование 7-разрядных кодов АЦПУ в 4-разрядные двоично-десятичные коды
144	преобразование 4-разрядных двоично-десятичных кодов в 7-разрядные коды АЦПУ .
145	перевод отредактированного по виду числа в 2 с/с
146	пересылка строки в строчную переменную .
147	перевод числа из 2 с/с в 10 с/с с редактированием по виду
150	числовой ввод с перфокарт
152	числовой вывод на перфокарты
154	текстовой ввод с перфокарт
155	текстовой вывод на перфокарты
156	сравнение строчных величин
157	числовой ввод с перфоленты в МОЗУ-2
160	числовой ввод с перфокарт в МОЗУ-2

Примечание. 2 с/с—двоичная система счисления; 10 с/с—десятичная система счисления.

3.10.1. Остановы в стандартных программах

На СчАК высвечивается один из адресов рабочего поля ИС, а на СМ—информация:

—0000	№ СП № останова	адрес команды

где № СП — номер СП, в которой произошел останов (9 разрядов), № останова — номер останова в СП (3 разряда).

Остановы в СП, приведенные ниже, перечислены в соответствии с номерами, высвечивающимися на СМ по первому адресу.

Список остановов в стандартных программах БСП

Содер-жимое СМ	Причина останова	Рекомендация
0251	$\sqrt{x}$: при $x \leq -0$	Устранить ошибку в программе
0261	e^x; при $x \geq 63\ ln\ 2$	Устранить ошибку в программе
0271	lnx; при $x \leq 0$	
0621	В переводе участвует очень большое число	
0711	Длина внутренней величины не равна длине внешней	Привести в соответствие описания внутренней и внешней величин
0712	Несравнение контрольной суммы с кодом—7777 7777 7777 при контрольном чтении информации с МЛ .	«Пуск» для повторной записи
0671	Длина внутренней величины не равна длине внешней	Привести в соответствие описания внутренней и внешней величин
0672	Несравнение контрольной суммы с кодом —7777 7777 7777 при чтении информации с МЛ	«Пуск» для повторного чтения
0751	В переводе участвует очень большое число	
0761	»	
1221	Не найден код символа текстовой информации в словаре перекодировки. Ненайденный код высвечивается на РГ1.	Исправить текстовую информацию
1222	Несравнение контрольных сумм при вводе текстовой информации с ПФЛ	«Пуск» для повторного ввода
1223	Вводимый массив текстовой информации превышает 2000_8 ячеек	
1224	В повторителе символа текстовой информации имеется символ «не цифра»	Для обнаружения ошибки можно использовать СП-22 для распечатки вводимых символов, изменив номер подключаемой СП-122 рабочей программы на 22 и прорешив задачу заново
1225	В повторителе символа текстовой информации нет закрывающей или открывающей скобки-	
1231	Не найден код АЦПУ в словаре перекодировки текстовой информации	
1232	Выводимый массив текстовой информации превышает 2000_8 ячеек	
1261	Несравнение контрольных сумм при вводе с ПФЛ числовой информации	«Пуск» да для повторного ввода
1262	Вводимый с ПФЛ числовой массив превышает 2000_8 ячеек	См. разд. 3.11. 2.
1361	СП перевода из системы с плавающей запятой в систему с фиксированной запятой. Переводится очень большое число	
1371	Потеря 8-ми младших разрядов в преобразуемом целом числе. Максимальная ошибка равна 255	«пуск» для дальнейшей работы программы
1401	Деление на 0 в программе деления целых чисел	
1451	Неправильно указан вид	
1471	Неправильно указан вид	
1501	В кармане перфокарточного ввода карт меньше, чем требуется по программе	
1521	Не работает устройство ввода. Числовая информация, считываемая с контрольных щеток, не сравнивается с содержимым ячеек МОЗУ	
1551	Не работает устройство вывода. Текстовая информация, считываемая с контрольных щеток, не сравнивается с содержимым ячеек МОЗУ	
1571	Вводимый с ПФЛ числовой массив, превышает длину, заданную в описании	
1572	Несравнение контрольных сумм при вводе с ПФЛ числового массива	«Пуск» для повторного ввода

П р и м е ч а н и е . Для СП-22 действительны остановы: 1221, 1222 и 1223

3.10.2. Включение в библиотеку стандартных программ (БСП) новых процедур-кодов

Список зарезервированных в трансляторе спецификаций процедур-кодов и описания соответствующих стандартных программ приведены в описании входного языка (см. разд. 1.4.7 и 2.17). Кроме того, допускается включение в БСП новых СП, записанных в кодах ЭВМ «Минск-22» и оформленных специальным образом.

3.10.2.1. ОБЩИЕ ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ СП

Правила оформления СП совпадают с соответствующими требованиями к оформлению стандартных программ для ЭВМ «Минск-22», перечисленными в описании библиотеки стандартных программ для ЭВМ «Минск-2»[10].

1. СП должна быть составлена в действительных адресах, начиная с адреса 7000. СП должна занимать не более 1000_8 ячеек (7000—7777).

2 . СП должна состоять из двух частей: перерабатываемой и неперерабатываемой. Каждое слово (команда, константа) будет отнесено к перерабатываемой или к неперерабатываемой части в зависимости от того, изменяется или не изменяется адресная часть соответствующего машинного слова при настройке СП на конкретное место памяти. Признаком того, что адрес является изменяемым (в перерабатываемой части), служит его старшая цифра 7. Таким образом, адреса, начиная с 7000, играют роль символических адресов. Адреса ряда команд перерабатываемой части (например, условные числа в команде печати, ленточные адреса в командах обращения к МЛ) не подвергаются изменению даже в тех случаях, когда они формально являются символическими, т. е. начинаются с цифры 7. Структуру СП можно представить в следующем виде:

перерабатываемая. часть
неперерабатываемая часть
свободная ячейка
информационная ячейка

3. СП может быть простой или сложной. СП, которая в процессе своей работы не обращается ни к внешним блокам, ни к другим стандартным программам, называется простой. В противном случае СП называется сложной.

4. Последняя ячейка .неперерабатываемой части является информационной для данной СП. В 12-ти старших (левых) разрядах этой ячейки указывается длина перерабатываемой части:

+ -	длина перерабатываемой части		.

Эта ячейка входит в счет длины СП. Кроме того, для сложных СП в младших (правых) 12-ти разрядах информационной ячейки должна быть указана общая длина подпрограммы с учетом длин тех блоков ,и СП, к которым будет обращаться данная сложная СП в процессе своей работы.

+ -	длина перерабатываемой части		общая длина блоков сложной СП.

Предпоследняя, ячейка, предназначенная для контрольной суммы СП, остается свободной.

5. Во время работы сложной СП допускается три уровня вхождения СП друг в друга:

рабочая программа
0-й уровень — *сложная СП* 1-й уровень. — *сложная СП* 2-й уровень — *простая СП* 3-й уровень

Количество СП на каждом уровне не ограничено.

6. Все стандартные программы используют общий массив рабочих ячеек (00020—02077). Допускается использование 15 индексных ячеек 00001—00015. Во время работы сложной СП при переходе на более глубокий уровень (т. е. при обращении данной сложной СП к внутренней) все указанные индексные ячейки запоминаются и соответственно восстанавливаются интерпретирующей системой в тот момент, когда внутренняя СП заканчивает работу. ИС не использует и не сохраняет рабочие ячейки стандартных программ. Аргументы засылаются, начиная с ячейки 00030.

Вход в СП осуществляется через ее первую команду, выход — по ячейке 00017, выход из ИС в рабочую программу по ячейке — 00016. Команда обращения к ИС — 310021500017, далее указывается номер стандартной программы,

3.10.2.2. ВКЛЮЧЕНИЕ ПРОЦЕДУР-КОДОВ В БСП

В комплект обслуживающих программ входит программа включения стандартных программ в БСП транслятора (см. разд. 3.8.5).

3.10.2.3. ФАКТИЧЕСКИЕ ПАРАМЕТРЫ ПРОЦЕДУР-КОДОВ

Оператор процедуры-кода реализуется в рабочей программе как обращение к соответствующей СП с предварительной засылкой в определенные рабочие ячейки информации о фактических параметрах. Следует помнить, что во всех случаях в качестве информации о фактических параметрах засылаются координаты расположения их в памяти, но не значения. Указанные координаты являются величинами типа **целый**. Фактические параметры оператора процедуры-кода размещаются в МОЗУ-2; исключение составляют константы, размещаемые вместе с рабочей программой в МОЗУ-1. При пересылке информации о фактическом параметре признаком того, что величина находится в МОЗУ-1, является единица (минус) в знаковом разряде ячейки, содержащей начальный адрес указанного фактического параметра.

1) В качестве информации о простой переменной, не имеющей в описании указания вида, а также о переменной с индексами, являющейся элементом массива без указания вида, засылается адрес указанного фактического параметра в одну ячейку.

2) В качестве информации по групповым параметрам засылаются начальный адрес группового параметра—в первую ячейку, а количество ячеек, отведенных под групповой параметр, — во вторую ячейку.

3) В качестве информации по простым переменным типа **целый** или **вещественный,** имеющим согласно описанию указание вида, засылается адрес переменной в первую ячейку. При этом следует иметь в виду, что значение переменной, находящееся по указанному адресу, представлено либо в двоично-десятичном коде (если в указании вида используются 9 позиций), либо в двоичном коде (в случае использования 1 позиции) и записано, начиная с левого конца ячейки. Во вторую ячейку заносится указание вида, которое также располагается, начиная с левого конца ячейки. Значение переменной и указание вида кодируются следующим образом:

символ	*двоичный код*
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
+	1010
-	1011
(	1100
)	1101
	1110
$_{10}$	1111

4) По переменной с индексами, являющейся элементом массива с указанием вида, а также по переменной типа **строчный** пересылаются четыре аргумента. Первый аргумент — это номер разряда без единицы относительно начала МОЗУ-2, с которого располагается переменная. Второй аргумент — количество разрядов, отведенных для данной переменной; третий аргумент — тип переменной, записанный в 11-м и 12-м разрядах; второй и третий заносятся в одну ячейку. Четвертый аргумент — это указание вида. Значение переменной и указание вида кодируется так, как указано выше (см. п. 3). По переменной типа **строчный** пересылается указание вида, равное нулю. Кодировка типов следующая: **целый —01, вещественный** — 10, **строчный**—11 (в двоичном коде).

Начальный разряд можно пересчитать в адрес, разделив номер его (указанный в первой из трех ячеек) на 36ю для переменных типа целый или вещественный и на 35_{10} для переменных типа строчный. Второй аргумент можно пересчитать в количество занимаемых данной переменной ячеек. Для этой цели используется специальная программа приводимая ниже.

5) В качестве информации по строчным константам (строкам) засылаются:

в первую ячейку — начальный адрес в МОЗУ-1 (в знаковом разряде минус); во вторую ячейку — количество занимаемых ячеек без единицы; в третью ячейку — количество незанятых справа разрядов в последней занимаемой константой ячейке.

6) В качестве информации по числовой константе пересылается сама константа.

Указанная информация о фактических параметрах засылается подряд, начиная с ячейки 00030, в порядке записи соответствующих фактических параметров в операторе процедуры-кода.

Оператор процедуры-кода оформляется в соответствии с синтаксисом (см. описание входного языка), причем, в качестве спецификации процедуры-кода в кавычках указывается номер соответствующей СП, а первый фактический параметр (целое без знака) является фиктивным и указывает число собственно фактических параметров, перечисленных вслед за ним.

Программа пересчета разрядов в адреса:

Обращение к СП осуществляется командой —31 00 2100 2120

Аргументами программы являются:

а) номер разряда относительно начала МОЗУ-2, с которого располагается переменная — в ячейке 00051;

б) количество разрядов, отведенных под переменную в разрядах 13—36 и тип переменной — в 11-м и 12-м разрядах ячейки 00052.

Результаты работы программы находятся в ячейках:

в 00050—тип переменной (в младших разрядах в указанной выше кодировке);

в 00051— начальный адрес переменной в МОЗУ-2;

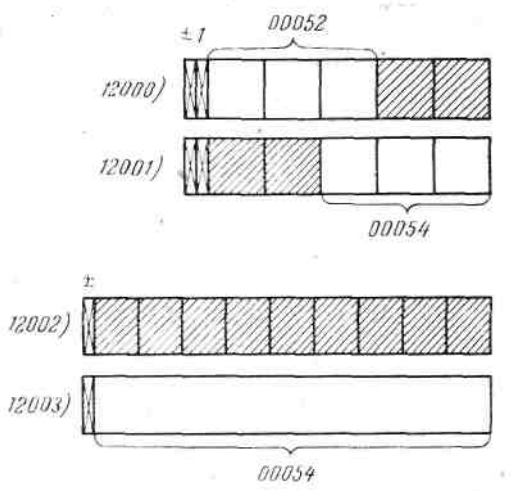
в 00052 — номер разряда без единицы, начиная с которого располагается переменная в указанной ячейке, т. е. количество разрядов слева, не занятых переменной.

в 00053—количество ячеек без единицы, занимаемых данной переменной;

в 00054 — количество разрядов справа в последней ячейке, не занятых данной переменной (36_{10} или 35_{10} — за вычетом количества разрядов, занимаемых данной переменной в последней ячейке). Число, находящееся в ячейке 00054, в частном случае может быть равно 36_{10} (или 35_{10}) для переменных типа **строчный.** Это значит, что в ячейке 00053 указано истинное число ячеек, отведенных под переменную.

На рисунке заштрихована обрабатываемая информация.

Если общее число ячеек (по всем фактическим параметрам оператора процедуры-кода) превышает 10_8 (ячейки 00030÷00037) и если в процессе работы такой стандартной программы необходимо использовать программу пересчета разрядов в адреса, приведенную выше, то перед обращением к программе пересчета, внутри включаемой СП, содержимое ячеек, начиная с 00040, необходимо переслать на рабочее поле самой стандартной программы, поскольку программа пересчета использует в качестве рабочих ячейки 00040—00044 и 00050—00054. (Под рабочим полем СП понимаются ячейки, отведенные под ее перерабатываемую и неперерабатываемую части.)




```
00050)  +00    00    0000    0003
    1)  +00    00    0000    2000
    2)  +00    00    0000    0025
    3)  +00    00    0000    0001
    4)  +00    00    0000    0025
```

```
00050)  +00    00    0000    0002
    1)  +00    00    0000    2002
    2)  +00    00    0000    0000
    3)  +00    00    0000    0001
    4)  +00    00    0000    0044
```

Пример.

```
начало  целый  А, С вид 99;
   массив В вид 99.9(10)+9, Д, Е [1:N];. . .
  КОД ('00000040', 7, А, В [I], С, Д, Е [К], 6, 'КОНЕЦ'); ...
конец
```

Команды обращения к СП-0040 (в кодах ЭВМ «Минск-22») будут иметь следующий вид:

```
04000) —10  00  7723  0030      в 07723 записан адрес переменной А
    1) —10  40  0203  0031 .    в  10203 находится  начальный разряд В  [I]
    2) —10  40  0204  0032      в  10204 находится количество разрядов, занимаемых В [I], и тип
    3) —10  00  7720  0033      в 07720 указание вида переменной В [I]
    4) —10  00  7724  0034      в 07724 адрес переменной С
    5) —10  00  7717  0035                в 07717 указание вида переменной С
    6) —10  40  7760  0036      в 17760 начальный адрес массива Д
    7) —10  40  7761  0037      в 17761 количество ячеек в массиве Д
04010) —10  40  0205  0040      в 10205 адрес К-го элемента массива Е
    1) —10  00  7715  0041      в 07715 константа 6
    2) —11  00  7712  0042      в 07712 начальный адрес строки 'КОНЕЦ1'
    3) — 10  00  7711  0043     в 07711 количество ячеек без единицы, занимаемых строкой  'КОНЕЦ1'
    4) — 10  00  7710  0044     в 07710 количество незанятых справа разрядов в последней ячейке
    5) — 31  00  2150  0017     обращение к ИС для вызова СП 40
    6) +  00  00  0000  0040
07710)+ 00  00  0000  0034
    1)+ 00 00   0000  0001
    2)+ 00  00  0000  7713
    3) +12  45  6265  1266      строка 'КОНЕЦ1' в семиразрядной кодировке
    4) +00 20   0000   0000     АЦПУ= 128
07715) +00 00   0000   0006     константа 6
07717) +46 20  0000   0000      указание вида 99 (см. кодировку)
07720)+46 36  4772   4400       указание вида 99.9(10)+9
07723) +00 00  0000   2005      адрес переменной А в МОЗУ=2
07724)+00  00 0000   2006       адрес переменной С в МОЗУ=2
```

Пусть в момент выполнения участка программы 04000—04016 в МОЗУ-2 находилась следующая информация:

```
10203)+00     00    0002    2115
    4)+00     02    0000    0034
    5)+00     00    0000    0512
17760)+00     00    0000    0472
    1)+00     00    0000    0170
```

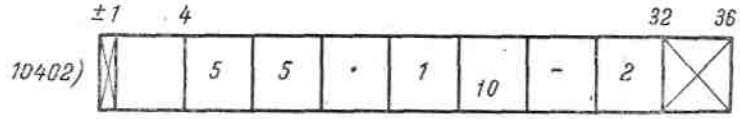
Значение переменной А находится в ячейке 12005. Значение переменной В [I] находится в МОЗУ-2, начиная с разряда 22115, и занимает 34_8 разряда (28_{10}). Если обратиться к программе пересчета разрядов в адреса:

```
-10 00  0031  0051
-10 00  0032  0052
-31 00  2100  2120,
```

т. е. заслав первый аргумент (22115) в ячейку 00051 и второй аргумент (34) в ячейку 00052, получим в результате работы этой программы:.

```
00050)+00    00    0000  0002
   1)+00     00    0000  0402
   2)+00     00    0000  0004
   3)+00     00    0000  0000
   4)+00     00    0000  0004
```

Это значит, что значение В [1] (55.1_{10}—2) для данного конкретного значения 1 находится в ячейке 10402, начиная с 5-го разряда и занимает неполную ячейку; четыре последних (правых) разряда этой ячейки не заняты.



Значение переменной С находится в ячейке 12006. Массив Д размещается, начиная с ячейки 10472, и занимает 170_8 ячеек. Значение переменной Е [К] для данного конкретного К находится в ячейке 10512. Константа 6 находится в ячейке 07715.

3.11. НЕКОТОРЫЕ ПРОЦЕДУРЫ-КОДЫ, ВКЛЮЧЕННЫЕ В БСП ТРАНСЛЯТОРА СТ-3

В этом разделе приводятся описания некоторых процедур-кодов, составленных в кодах ЭВМ «Минск-22», оформленных согласно инструкции, данной в разделе 3.10.2. и включенных в БСП транслятора.

3.11.1. Процедура-код выполнения операций отношения над величинами типа СТРОЧНЫЙ (СП-156)

Обращение к СП-156 содержит четыре собственно фактических параметра. Первый параметр — целое без знака, задающее в закодированной форме операцию отношения.

Кодировка операций отношения следующая:

<	1
≤	2
=	3
≠	4
>	5
≥	6

Второй и третий фактические параметры — операнды отношения. В качестве операндов отношения допускается использование переменных типа **строчный** и строк.

Таблица старшинства строчных символов

Символ	Код АЦПУ	Символ	Код АЦПУ	Символ	Код АЦПУ
0	000	‘	032	Ф	064
1	001	,	033	Х	065
2	002	≠	034	Ц	066
3	003	<	035	Ч	067
4	004	>	036	Ш	070
5	005	:	037	Щ	071
6	006	А	040	Ы	072
7	007	Б	041	Ь	073
8	010	В	042	Э	074
9	011	Г	043	Ю	075
+	012	Д	044	Я	076
—	013	Е	045	D	077
/	014	Ж	046	F	100
,	015	З	047	G	101
.	016	И	050	I	102
пробел	017	Й	051	J	103
10	020	К	052	L	104
↑	021	Л	053	N	105
(	022	М	054	Q	106
)	023	Н	055	R	107
×	024	О	056	S	110
=	025	П	057	U	111
;	026	Р	060	V	112
[	027	С	061	W	113
]	030	Т	062	—Z (надчеркивание)	114
*	031	У	063		115

Четвертый фактический параметр — простая переменная типа **целый,** не имеющая указания вида. Она используется для засылки результата операции (1—для «истина» и 0 — для «ложь»).

Сравнение операндов выполняется слева направо символ за символом. Старшинство строчных символов задается таблицей (см. стр. 80).

Более короткое слово считается дополненным справа пробелами (код 17) до выравнивания длин.

Пример:

Расположить элементы массива А строчных переменных в лексикографическом порядке.

```
начало  строчный  массив А вид С (15)  [1:10];
    строчный РЯ вид С (15), В вид С (15);
    целый К, J, Р;
    КОД ('ВВОДЛ_С', А);
    для К: = 1 шаг 1 до 10 цикл
    начало РЯ:=А [К];
        для J:=K+1 шаг 1 до 10 цикл
        начало КОД ('00000156', 4, 1, А [J], РЯ, Р);
          если Р=1 то начало
          В:=РЯ;   РЯ:=А [J];   А[J]:=В конец
        конец; А [К]:=РЯ
    конец;
   для К: = 1 шаг 1 до 10 цикл
   КОД ('ПЧ_АЦПУ, А [К])
конец
```

Обращение к СП-156 в данном примере оформляется следующим образом:

```
КОД ('00000156', 4, 1, А [J], РЯ, Р);
```

В записи после номера стандартной программы указано количество параметров (4), затем вид сравнения «меньше» (1), далее строчные величины А[J] и РЯ, над которыми выполняется указанная операция; Р — результат сравнения, принимающий значение, равное 1, если отношение, задаваемое в обращении, истинно, и 0, — если оно ложно.

Обращение к СП-156 в составленной транслятором программе будет записано следующим образом:

```
07200) —10    00  7721   0030      1-й аргумент
    1) —10    40  0205   0031   }
    2) —10    00  7712   0032   }  2-й аргумент
    3) —10    00  7717   0033   }
    4)—10     40  0206   0034   }
    5)—10     00  7712   0035   }  3-й аргумент
    6)—10     00  7717   0036   }
    7)—10     00  7711   0037      4-й аргумент
 7210)—31     00  2150   0017   }  обращение к СП
    1)  +00   00  0000   0156   }
07711) + 00   00  0000    7751    адрес Р в МОЗУ-2
07712) + 00   03  0000    0151    длина строчной величины в разрядах и тип переменной
07717) + 00   00  0000    0000    вид по переменным типа строчный условно равен 0
07721) + 00   00  0000    0001    код операции отношения «меньше»
```

В ячейке 10205 записан номер разряда без единицы, начиная **с** которого располагается элемент А[J]; в ячейке 10206 — номер разряда без единицы, начиная с которого располагается РЯ.

Вход **СТРУКТУРА СП-156**

Расчет адреса А[J], обращение: —10 00 0031 0051 —10 00 0032 0052 —31 00 2100 2120
Расчет адреса РЯ, обращение: —10 00 0034 0051 —10 00 0035 0052 —31 00 2100 2120
Сравнение строчных величин и определение Р
—10 00 0037 0001 —10 21 Р 0000 Выход

Результат сравнения засылается перед выходом из СП-156 по адресу, отведенному Р при распределении памяти и передаваемому в стандартную программу в качестве аргумента.

При работе программы, приведенной в примере, были получены следующие результаты:

информация		*результаты*	
_Сидоров _ _	1942 _	_Андреев _ _	1940_
_Андреев _ _	1940_	_Андровская	1941
_Иванов _ _ _	1941_	_Иваненко_	1936_
_Рощин _ _ _ _	1939_	_Иванихин_	1930_
Иващенко	1940_	_Иванов _ _ _	1941_
_Петров _ _ _	1940_	_Иванов _ _ _	1950_
Иванихин	1930_	_Иващенко_	1940_
_Андровская	1941	_Петров_ _ _	1940_
Иваненко	1936_	_Рощин _ _ _ _	1939_
Иванов _ _	1950_	_Сидоров_ _	1942_

3.11.2. Процедура-код ввода в МОЗУ-2 числовой информации с перфоленты (СП-157)

Стандартная программа рассчитана. на ввод с перфоленты одного массива числовой информации. Длина вводимого массива определяется описанием этого массива и не может превышать 7400_8 ячеек. Информация должна быть отперфорирована без адресных кодов. В том случае, если длина массива на перфоленте превышает заданную в описании, происходит останов.

Перевод чисел из десятичной системы в двоичную не предусмотрен; чистка памяти **в** МОЗУ-2, отведенной под данный массив стандартной программой, производится перед вводом массива.

Обращение к стандартной программе ввода имеет вид:

КОД ('00000157', 1, <групповой параметр>)

Например:

КОД ('00000157', 1, A. B. [J])

3.11.3. Процедура-код ввода в МОЗУ-2 числовой информации с перфокарт (СП-160)

Стандартная программа рассчитана на ввод **с** перфокарт одного массива числовой информации любой длины (не более 7400_8 кодов). При обращении указываются параметры, первый из которых— вид вводимой информации (8 с/с — признак 0, 10 с/с — признак 1) и второй — идентификатор вводимого массива (групповой параметр).

Например:

КОД ('00000160', 2,0, A. M)

Эта запись означает ввод с перфокарт массива А.М., отперфорированного в восьмеричной системе счисления.

Перевод чисел из десятичной системы в двоичную не предусмотрен. Программа допускает ввод меньшего количества перфокарт по сравнению с рассчитанным из описания. При этом следует помнить, что в кармане вводного устройства должен быть только один этот массив. Память в МОЗУ-2, отведенная под данный массив, перед вводом не чистится. Рекомендуется в конце вводимого массива перфорировать признак конца массива, который следует проверять во время обработки.

3.11.4. Процедура-код записи информации на магнитную ленту с дополнением ее контрольной суммы до кода —7777 7777 7777 (СП-71)

Обращение к процедуре-коду имеет вид:

КОД ('00000071', 2, <групповой параметр>, <групповой параметр>)

Первым фактическим параметром является аргумент-отправитель, т. е. внутренняя величина, а вторым — аргумент-получатель, т. е. внешняя величина.

Значение внутренней величины записывается на магнитную ленту с дополнением контрольной суммы информации до кода —7777 7777 7777.

В МОЗУ это дополнение заносится в последнюю ячейку памяти, отведенной под внутреннюю величину согласно распределению памяти.

Это следует, учитывать при описании внутренней и внешней величин, используемых в операторе процедуры-кода с данной спецификацией.

Пример 1.

```
...массив А, Я00000000 [1:51];
КОД ('00000071', 2, А, Я00000000); ...
```

Массивы А и Я 0000 0000 содержат по 51 элементу, причем,. 51-й элемент отведен для дополнения контрольной суммы до кода —7777 7777 7777

Пример 2.

```
...составной Я00000000; А, M×N + - уровень;
составной К; массив В [1:M, 1:N];
целый ДКС уровень; ...
КОД ('00000071', 2, К, Я00000000); ...
```

Дополнение контрольной суммы до кода —7777 7777 7777 в данном примере запишется в ячейку,

отведенную для переменной ДКС.

Необходимым условием при обращении к данной процедуре-коду является равенство длин (по количеству ячеек) внешней и внутренней величин, используемых в качестве фактических параметров.

В случае неравенства длин происходит останов (см. список остановов в БСП транслятора).

3.11.5. Процедура-код чтения с магнитной ленты информации, контрольная сумма которой дополнена до кода —7777 7777 7777 (СП-67)

Обращение к процедуре-коду имеет вид:

КОД ('00000067', 2, <групповой параметр>, <групповой параметр>)

Первым фактическим параметром является аргумент-отправитель, т. е. внешняя величина, а вторым — аргумент-получатель, т. е. внутренняя величина.

Считается, что внешняя величина записана на магнитную ленту с использованием процедуры-кода со спецификацией '00000071', т. е. с дополнением контрольной суммы информации до кода —7777 7777 7777.

В результате чтения внешней величины в МОЗУ дополнение контрольной суммы до кода —7777 7777 7777 будет находиться в последней ячейке памяти, отведенной под внутреннюю величину.

Это следует учитывать при описании внутренней и внешней величин, используемых в операторе процедуры-кода с данной спецификацией.

Пример 1.

```
... массив А, Я00001000 [1:51];
КОД ('00000067', 2, Я00001000, А);...
```

Массивы А и Я00001000 содержат по 51 элементу, причем 51-й элемент отведен под дополнение контрольной суммы до кода —7777 77777777.

Пример 2.

```
... составной Я00001000; А, М×N+1 уровень;
составной К; массив В [1:М, 1:N];
целый ДКС уровень;
КОД ('00000067', 2, Я00001000, К); . . .
```

Дополнение контрольной суммы до кода —7777 7777 7777 в данном примере запишется в ячейку, отведенную для переменной ДКС.

Необходимым условием при обращении к данной процедуре-коду является равенство длин (по количеству ячеек) внешней и внутренней величин, используемых в качестве фактических параметров.

В случае неравенства длин происходит останов (см. список остановов БСП транслятора).

3.11.6. Процедура-код подвода зоны НМЛ (СП-72)

Обращение к процедуре-коду имеет вид:

КОД ('00000072', 1, <групповой параметр>)

где фактическим параметром является внешняя величина, зону которой требуется подвести.

Пример

```
... массив Я00004000 [1: 100]; ...
КОД ('00000072', 1, Я00004000).
```

3.12. ИНСТРУКЦИИ ДЛЯ ОПЕРАТОРОВ ПО РАБОТЕ ЗА ПУЛЬТОМ

3.12.1. Инструкция № 1. Трансляция программ

1. Поставить магнитную ленту с программами транслятора на 02 ЛПМ, рабочую ленту на 01 ЛПМ.
2. Поставить кольцо с транслируемой АЛГЭМ-программой на фотоввод.
3. Включить АЦПУ, фотоввод, ключи, указанные в задании разработчика программы.
4. Занести с пульта и выполнить в режиме «автомат» следующие команды:

00001)—47	00	0200	4000
00002)—45	00	2000	0001
00003)—30	00	0001	0000

5. Останов СчАК—13673 — конец работы транслятора.

Положение ключей

включен ключ 1—выдача на АЦПУ распределения памяти,
включен ключ 2 — вывод на перфоратор 1 программы в записи на АЛГЭМе,
включен ключ 4 — вывод на перфоратор № 1 готовой программы,
включен ключ 10 — запись готовой программы на магнитную ленту, (необходимо на наборе кода указать команду: —4700 *pql),*
включен ключ 20 — настройка рабочей программы на ввод — вывод вещественных чисел в режиме «с мантиссой и порядком»,
включен ключ 40 — вывод на печать таблицы меток.

6. См. также «Список остановов транслятора». (разд. 3.7).

3.12.2. Инструкция № 2. Работа составленных транслятором программ

Вариант 1. Решение задачи сразу после трансляции (программа находится в МОЗУ)

1. Включить все устройства, необходимые при работе программы.
2. Передать управление по адресу 07727 в режиме «автомат».
3. СчАК 07771 —конец работы.

Вариант 2. Программа при трансляции была отперфорирована на ПЛ.

Поставить ПЛ программы на фотоввод и ввести программу. Если контрольная сумма ввода —7777 7777 7777, то выполнить пункты 1, 2, 3 варианта 1. В противном случае повторить ввод программы.

Вариант 3. Программа при трансляции была записана на магнитную ленту (МЛ).

1. Поставить МЛ с программой на ЛПМ, указанный в задании программиста.
2. Занести и выполнить в режиме «цикл» следующие команды:

00001)—47	00	*pql*	
00002)—45	00	4000	4000
00003)—30	00	0001	0000

Если контрольная сумма при считывании равна —7777 7777 7777, то выполнить пункты 1, 2, 3 варианта 1. В противном случае повторить считывание.

Примечание 1. *p, q, l,* задаются в инструкции разработчиком программы.
2. См. также «Описание работы составленных транслятором программ» (3.9).

ЛИТЕРАТУРА

1. Алгоритмический язык АЛГОЛ-60. Пересмотренное сообщение. Перевод с англ. под ред. А. П. Ершова, С. С. Лаврова я М. Рура-Бура. Изд-во «Мир», 1965.
2. Л е д л и Р. Программирование и использование вычислительных машин. Пер. с англ. под ред. А.. И. Китова. Изд-во «Мир», 1966.
3. С. С. Л а в р о в . Универсальный язык программирования АЛГОЛ-60.
4. А. И. К и т о в . Программирование информационно-логических задач. Изд-во «Советское радио», 1967 г.
5. Ф. Ф. Шиллер. О контроле правильности программы, записанной на алгоритмическом языке, в условиях применения синтаксического транслятора. Сб. Цифровая вычислительная техника и программирование», вып. 5, 1969.
6. Ю. П. Кирюхин. Алгоритмы составителя таблиц синтаксического транслятора СТ-3. Сб. «Цифровая вычислительная техника и программирование», вып. 5, 1969.
7. П. Инге р м а н . Синтаксически ориентированный транслятор. Перевод с англ. под ред. А. И. Китова. Изд-во «Мир», 1969.
8. E. I. I r o n s . A structure and use of the syntax directed compiler. Annual Review in automatic programming, v. 3, 1962.
9. Машина вычислительная цифровая универсальная «Минск-2». Инструкция по эксплуатации, ч. II. Инструкция по программированию. 1Ф.320.003, ред. 2—64, [Минск], 1964, 42с.
10. Библиотека стандартных программ для ЦВМ «Минск-2». М., 1963. 229с. СНХ БССР. Управление электротехнической и приборостроительной промышленности. Научно-исследовательский институт по проектированию вычислительных центров и систем экономической информации. ЦСУ СССР. Перед загл. авт.: Г. К. Столяров, М. Е. Неменман, Э. В. Ковалевич и др.