МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. ЛОМОНОСОВА

Вычислительный центр
Е.А.Жоголев, Н.Б.Лебедева

СИМПОЛИЗ 64 — Язык для программирования в символических обозначениях

Серия:
Математическое обслуживание
машины «Сетунь»

Под общей редакцией Е.А.Жоголева
Выпуск 10

Издательство Московского
Университета 1965

Содержание

Введение.

В Вычислительном центре МГУ в 1963-64 годах разработана для машины «Сетунь» система автоматического кодирования, получившая название ПОЛИЗ. Основу этой системы составляет ПОЛьская Инверсная Запись (ПОЛИЗ) алгоритмов, основные идеи которой были предложены польским математиком Лукашевичем для бесскобочной записи формул. В данной работе описывается входной язык этой системы. Этот язык представляет собой польскую инверсную запись в символических обозначениях, для которых используются буквы, цифры и другие воспринимаемые машиной «Сетунь» знаки.

Программа, задаваемая на этом входном языке, не переводится на язык машины, а лишь представляется в троичном коде с сохранением самой польской инверсной записи. Такое кодирование осуществляется автоматически с помощью специальной программы. Полученная в результате этого программа выполняется в режиме интерпретации.

Таким образом, система автоматического кодирования ПОЛИЗ слагается из следующих компонент:

- Входного языка, представляющего собой польскую инверсную запись в символических обозначениях (СИМПОЛИЗ);

- Внутреннего языка, представляющего собой польскую инверсную запись в троичном коде, т.е. интерпретируемую польскую инверсную запись (ИНПОЛИЗ);

- Программы автоматического кодирования (АВТОКОДИР ПОЛИЗ);

- Операционной системы ПОЛИЗ, состоящей из

  - Интерпретатора ПОЛИЗ,

  - Библиотеки подпрограмм, реализующих элементарные операции системы.

Отдельные компоненты системы автоматического кодирования могут выступать в разных вариантах, придавая системе в целом ту или иную направленность. Необходимость варьирования по крайней мере набора элементарных операций системы (а тем самым и библиотеки подпрограмм) не вызывает сомнений в связи со сравнительно небольшой емкостью всей памяти машины «Сетунь». Действительно, библиотека подпрограмм для достаточно полного набора элементарных операций системы будет занимать слишком много места в памяти или даже и совсем в ней не поместиться, а любой её «сокращенный» вариант, естественно, будет ориентирован на какой-либо более узкий класс задач. Для задач же другого класса будет явно не хватать некоторых операций, а некоторые операции будут излишними. Поэтому в этих условиях целесообразно иметь несколько вариантов библиотеки подпрограмм, ориентированных на разные классы задач. Новые варианты других компонент могут возникнуть и в ре-

зультате совершенствования системы. В связи с этим в название каждой компоненты будет добавляться индекс (как правило, две цифры года его создания).

Наиболее чувствительным к изменениям компонент системы является входной язык. Однако принципиальное изменение стиля языка могут внести только интерпретатор и автокодир. Поэтому эти две компоненты мы предполагаем здесь зафиксированными: данный входной язык основывается на использовании интерпретатора ПОЛИЗ 63 [1] и автокодира ПОЛИЗ 64 [2]. С учетом сказанного приводимый здесь набор элементарных операций системы определяет входной язык СИМПОЛИЗ 64. Однако данное руководство остается в силе и для любого другого языка, отличающееся от СИМПОЛИЗа 64 только набором элементарных операций (с заменой, естественно, описаний самих этих операций). Основу библиотеки подпрограмм, входящей в данную систему, составляют подпрограммы, разработанные для ИП-3 [3,4], в которые были внесены очевидные изменения, связанные с настройкой их для работы с другим интерпретатором и, частично, с перекомпановкой.

Разработку данной системы стимулировал тот факт, что польская инверсная запись была принята в качестве выходного языка транслятора с АЛГОЛа. Это привело к существенному упрощению алгоритмов трансляции и в то же время интерпретация польской инверсной записи весьма просто осуществляется на ма-

шине «Сетунь» (интерпретатор ПОЛИЗ, проще, чем, например, ИП-3). Так как транслятор с АЛГОЛа предполагалось реализовать только для машин с удвоенной емкостью барабана, то было признано целесообразным на всех машинах использовать разрабатываемую операционную систему (рассчитанную первоначально только для обслуживания программ, полученных с помощью транслятора) в качестве основы системы автоматического кодирования. Безусловно, есть много полезного в том, что программы, получаемые в результате использования систем автоматического программирования разных уровней, будут представлены на одном и том же языке.

Использование режима полной интерпретации для выполнения таких программ приводит, естественно, к замедлению решения задачи, однако в данном случае это замедление сравнительно небольшое, в среднем в 2,5-3 раза по сравнению с использованием ИП-3. Некоторой компенсацией этому является то, что программы в ПОЛИЗе оказываются в 2-3 раза короче, чем в системе ИП-3. Но конечно главный выигрыш при использовании данной системы будет заключаться в упрощении процесса составления программ и значительном сокращении времени, необходимого на подготовку задачи к решению на машине (включая и сам процесс программирования).

В разработке данной системы автоматического кодирования принимали участие Бондаренко Н.В. (под-

программы ввода и вывода информации), Есакова Л.В. (подпрограммы, реализующие арифметические операции и вычисление элементарных функций), Жоголев Е.А. (интерпретатор ПОЛИЗ 63, некоторые подпрограммы), Кауфман В.Ш. (подпрограммы, реализующие логические операции и операции отношения), Лебедева Н.Б. (автокодир ПОЛИЗ 64), Рамиль Альварес Х. (некоторые подпрограммы, реализующие операции управления, а также дополнение к библиотеке, дающее некоторые средства отладки программ).

Входной язык, также как и другие компоненты системы, неоднократно обсуждался авторами этой системы, и каждый из них внёс в него свой вклад. Основные же его черты определяются возможностями, заложенными в интерпретаторе ПОЛИЗ 63 и автокодире ПОЛИЗ 64.

Разработка данной системы была выполнена под научным руководством Е.А.Жоголева.

1. Сущность польской инверсной записи.

Программа в ПОЛИЗе представляет собой информацию о последовательности операций, которую нужно выполнить для решения задачи, и об операндах (аргументах) каждой операции. Сущность ПОЛИЗа состоит как раз в том, что информация об операндах каждой операции должна быть задана до указания самой этой операции (при чтении записи слева — направо). Одна-

ко здесь понятия «информация об операнде» и «операция» требует пояснения.

Операнд — это то, над чем производится операция. Каждый операнд представляет некоторое значение (см.ниже). «Информация об операнде» может явно содержать это значение (и ничего больше), или иметь ссылку на это значение, т.е. Содержать лишь его наименование (адрес), или, наконец, содержать формулу для определения этого значения, представленную опять-таки в ПОЛИЗе (т.е. то, что мы пытаемся сейчас определить, примеры см. В П.1.1.).

Каждая «операция» может потреблять какое-то число значений (операндов), указанных ранее, а также вырабатывать какое-то число новых значений, которые будут использоваться в качестве операндов для последующих «операций». Набор таких «операций» строго фиксирован в каждом конкретном варианте ПОЛИЗа и каждая «операция» должна быть точно определена её описанием (т.е. она является результатом специального соглашения). В этом описании должно быть указано число операндов и результатов, их взаимный порядок следования и типы допустимых значений (см. ниже).

К моменту выполнения каждой операции значения всех её операндов должны быть уже определены. Но так как при определении значения операнда какой-либо операции может потребоваться выполнение другой

операции (если этот операнд задан формулой) и, следовательно, предварительного определения значений других операндов, то к моменту выполнения некоторых операций могут быть определены также и значения операндов последующих операций. Таким образом, перед выполнением каждой операции будет образована некоторая последовательность значений, которую будем пока называть «последовательностью выработанных значений».

Теперь сущность ПОЛИЗа можно сформулировать более точно следующим образом. Каждая выполняемая операция, указанная в программе в ПОЛИЗе, потребляет значения только из последовательности выработанных значений и именно столько последних из неё, сколько операндов (по определению) должно быть у данной операции, и подставляет вместо них, вообще говоря, другое число значений (а может быть и ни одного), являющихся результатами этой операции. Тем самым образуется новая последовательность со старой «головой» и новым «хвостом». Образование перед каждой выполняемой операцией такой последовательности выработанных значений, «хвост» который удовлетворял бы определению данной операции — это является одной из задач программирования в ПОЛИЗе.

На внутреннем языке польская инверсная запись представляет собой последовательность коротких машинных слов (обобщенных адресов), которые мы в

дальнейшем будем называть слогами записи. Слоги могут быть трех типов:

1) операциями,

2) значениями,

3) ссылками, т.е. наименованиями (адресами) тех значений или других слогов, которые подлежат использованию.

1.1. Примеры.

В нижеприводимых примерах значения представляются числами, ссылки — буквами, а операции — соответствующими знаками.

№№ примеров	Обычная запись	ПОЛИЗ
1.	$X+5$	$X\,5+$
2.	$3\cdot X+Y\cdot Z$	$3\,X\times Y\,Z\times+$
3.	$(U+10)/v$	$U\,10+V/$

В программах вычисляемые по формулам значения должны присваиваться каким-либо переменным. Поэтому среди операций ПОЛИЗа должна быть операция = («присвоить» или «положить равным»), а в качестве значения одного из её операндов должно быть наименование (адрес) этой переменной. Для отличия таких значений от соответствующих ссылок будем ограничивать значе-

ния — наименования закрывающей (слева) и открывающей (справа) круглыми скобками. Сравни, например:

ссылка: X
значение:)X(

Так как числа могут быть многозначными (см.пример 3), а также и наименования могут быть целыми словами, то для более четкого разделения слогов друг от друга будем в дальнейшем всегда записывать слоги в один столбец (по одному слогу в каждой строке).

№№ примеров	Обычная запись	ПОЛИЗ
4.	$Y=X^3-A$	)Y(X X x X x A – =

Из-за недостатка отдельных знаков некоторые из операций будем обозначать подчеркнутыми словами,

например, GO TO («перейти к»), ELSE («иначе») и т.д. Смысл всех этих операций, а также правила составления программы в ПОЛИЗе будут изложены ниже. Сейчас же мы ограничимся чисто формальным приведением еще двух примеров.

№№ примеров	Запись на АЛГОЛе	ПОЛИЗ
4.	go to	)BEGIN(GO TO
5.	if X=Y then Z:=0	X Y REQUAL)L(ELSE)Z(0 = L…

1.2. Магазин.

Реализация польской инверсной записи требует наличия в машине памяти нового типа — магазинной памяти или просто магазина. Действительно, для хранения последовательности выработанных значений требуется некоторая группа ячеек памяти. Эта последовательность в процессе выполнения программы будет

то возрастать, то уменьшаться и всегда будет потребляться одно или несколько самих поздних из выработанных значений. Поэтому для запоминания какого-либо значения или выборки его из памяти важно знать не его адрес, а его место в указанной последовательности, т.е. порядок его записи в память машины.

Магазином назовем память, в которой каждое записанное значение сохраняется только до его первого вызова и из которой каждый раз выбирается значение, записанное последним из имеющихся в этой памяти значений. Образно говоря, принцип, положенный в основу работы магазина, противоположен принципу работы обычной очереди: каждое вновь записанное в магазин значение как бы «встает без очереди» для потребления его последующими операциями. Отсюда следует порядок выборки, из магазина хранящихся в нем значений, в частности, значение, записанное в магазин первым, будет выбрано из него последним. Магазин является специальной памятью, помимо которой имеется еще и основная память с выборкой и записью по адресам. Значение, выбираемое из основной памяти, посылается в очередную свободную ячейку магазина (т.е. в ячейку, доступную в данный момент для записи). Все операции используют в качестве операндов значения, хранящиеся в магазине, а результаты также направляют в магазин. Поэтому такие

операции оказываются безадресными. Если для выполнения какой-либо операции требуется знать адрес в основной памяти (например, при присвоении какой-либо переменной нового значения, т.е. при необходимости записать какое-либо значение из магазина в основную память), то этот адрес может всегда задаваться в качестве операнда этой операции, т.е. содержаться в самом магазине.

Ячейку магазина, в которую в данный момент возможна запись, будем называть окном магазина. При записи значений в магазин или при выборке значений из магазина его окно перемещается в ту или другую сторону.

Машина «Сетунь» не имеет памяти магазинного типа. Поэтому она вводится программным путем с помощью интерпретатора ПОЛИЗ. Для данной системы, которую представляет язык СИМПОЛИЗ 64, магазин состоит из семи длинных ячеек. При этом еще в окно магазина всегда записывается выполняемый слог. Поэтому при накоплении в магазине семи значений происходит аварийный останов.

Программа в ПОЛИЗе выполняется нормально только в том случае, если в любой момент последовательность выработанных значений будет содержать меньше семи значений. Удовлетворение этого требования является задачей программирования (в СИМПОЛИЗе 64).

1.2.1. Пример работы магазина.

Ниже приведена диаграмма выполнения примера 4. Величины U_1, U_2,… U_7 обозначают содержимое соответствующих ячеек магазина, причем первоначально перед выполнением программы всегда в окне магазина оказывается величина U_7. Магазин переполняется, когда на место U_1 будет записано какое-либо значение. Предполагается, что перед выполнением программы примера 4 в окне магазина находится U_7, т.е. в магазине нет выработанных значений.

После выполнения последней операции (=) значение U_6 оказывается записанным в основную память по адресу, представляемому значением U_7, и после этого магазин снова не будет содержать никаких значений (придет в первоначальное состояние). Это типично для выполнения любой правильно составленной программы. Однако в результате ошибок в программе магазин может «засоряться» значениями, которые никогда не будут потребляться. Поэтому целесообразно «очищать» магазин, т.е. приводить его в первоначальное состояние, перед выполнением каждой программы (однако, в данном выпуске на этом мы останавливаться не будем, см. инструкцию по использованию системы в последующих выпусках).

Диаграмма выполнения примера 4.

Выполняемый слог	Последовательность выработанных значений в магазине						
	U_1	U_2	U_3	U_4	U_5	U_6	U_7
Перед выполнением программы							
)Y(							)Y(
X						X	)Y(
X					X	X	)Y(
x						X^2	)Y(
X					X	X^2	)Y(
x						X^3	)Y(
A					A	X^3	)Y(
-						X^3-A	)Y(
=							

1.3. Типы значений.

В данной системе операции производятся в основном над значениями четырех типов: вещественными, целыми, булевыми и адресами.

1.3.1. Вещественные значения.

Каждое вещественное значение представлено в машине длинным машинным словом, которое рассматривается как число с плавающей запятой. Форма изображения такого числа совпадает с формой, принятой в ИП-3 [3]. Мантисса числа содержит 13 троичных разрядов, что дает возможность вести вычисления, примерно с шестью верными десятичными знаками.
Троичный порядок числа p удовлетворяет неравенству:

$$|p| \leq 0 .$$

Это позволяет представлять числа, отличные от нуля, примерно, в диапазоне от 10^{-19} до 10^{19}. Кроме того, старший разряд длинного слова, представляющего такое значение, всегда в этом случае будет содержать нуль, что существенно используется в данной системе.

1.3.2. Целые значения.

Каждое целое значение представлено в машине коротким машинным словом, которое рассматривается как число с запятой, зафиксированной перед самым младшим разрядом. Таким образом, целое число n хранится в машине в виде:

$3\,n\,e_F$

где e_F обозначает единицу младшего разряда, или, другими словами, хранится в машине в масштабе 3^{-6} $(3\cdot n\,3^{-6})$. Отсюда следует, что в младшем разряде такого короткого слова всегда содержится нуль. Такой способ представления целых чисел связан с обеспечением простоты переадресации, так как предполагается, что в данной системе целые значения будут в основном использоваться для управления вычислительным процессом.

Все целые значения n должны удовлетворять неравенству:

$$|n| \le 1093 .$$

Поэтому старший разряд короткого слова, представляющего целое значение, также будет всегда содержать нуль.

1.3.3. Булевы значения.

Каждое булево значение представлено в машине коротким словом, причем истине (true) соответствует число e_F, а ложь (false) представляется нулем. Старший разряд такого короткого слова также будет всегда содержать нуль.

1.3.4. Адреса.

Каждый адрес представлен коротким словом, рассматриваемым в качестве обобщенного адреса, как это принято в ИП-2 и ИП-3 [5]. Только в данном случае старший разряд адреса всегда будет содержать нуль. По форме представления адреса могут отличаться от целых значений только содержимым младшего (девятого) разряда представляющего их слова (у адреса оно может иметь произвольное значение), но функционально они выполняют в алгоритме разные роли. К адресу A может быть добавлено (по машинной операции сложения) целое значение n и тогда результат

$$A+n$$

будет также адресом некоторой ячейки, отстоящей от ячейки A на n длинных ячеек.

1.3.5. Другие типы значений.

В данной системе или в её кодификациях могут использоваться и значения, имеющие совсем иной смысл (например, некоторый текст). Иногда это может быть введено по усмотрению автора программы, который может приспособить для их использования имеющийся в системе операции. Никаких дополнительных

операций для таких значений, кроме некоторых операций вводе и вывода (см. ниже) в данной системе не предусмотрено.

Однако значение любого типа, явно посылаемое в магазин, должно содержать нуль в старшем разряде представляющего его машинного слова. Остальные значения должны задаваться в магазине неявно их адресами.

1.4. Программа в ИНПОЛИЗе.

Как было сказано ранее программа на внутреннем языке (в ИНПОЛИЗе) представляет собой последовательность слогов (трех разных типов). Старший разряд слога является признаком его типа $П_ф$. Для слога — значения $П_ф=0$, для слога — операции $П_ф=-1$, а для слога-ссылки $П_ф=1$. Таким образом, в программе явно могут быть указаны только те значения, которые представляются короткими машинными словами. Младшие восемь разрядов слога-операции образуют обобщенный адрес начала соответствующей подпрограммы, реализующей данную операцию, а младшие восемь разрядов слога-ссылки — обобщенный адрес значения или другого слога, на которое производится данная ссылка.

2. Описание СИМПОЛИЗа 64.

2.1. Алфавит и элементы языка.

Программа в СИМПОЛИЗе 64 строится из цифр, букв и других знаков, которые могут быть введены в машину «Сетунь». Из этих знаков строятся элементы языка: целые без знака, идентификаторы и подчеркнутые слова. Элементы вместе с некоторыми знаками и являются теми «кирпичами», из которых строится «здание» программы и все её основные части,

2.1.1. Алфавит.

Цифры: 0 1 2 3 4 5 6 7 8 9;

Буквы: A B C D E F G H I J K L M N P Q R S T U V W X Y Z;

Другие знаки: + — x / = . ();

Подчеркивание: пч;

Этот символ позволяет вводить в машину и выводить из машины подчеркнутой любую последовательность из указанных выше знаков. При подаче этого символа на печать в нижней части соответствующей позиции (на бумаге) наносится горизонтальная черточка и блокируется переход к следующей позиции. Поэтому следующий выводимый знак будет отпечатан в той же позиции и тем самым оказывается подчеркнутым. При перфорации подчеркнутых слов перед каждым

подчеркнутым знаком (буквой) ставится (наносится) этот символ.

Управляющие комбинации: эти комбинации обеспечивают нужный формат записи (печати) в частности, разбиение записи на строки. Никакие сведения об этих комбинациях не требуются при написании программы в СИМПОЛИЗе 64. Однако при задании исходных данных (см. п.3.8) они могут существенно использоваться и поэтому там подробно описываются.

2.1.2. Целые без знака.

Целым без знака является любая последовательность цифр, имеющая обычный смысл. Однако в силу ограничений, накладываемых на целые значения целое без знака должно быть не больше 1093.

Примеры:

0
5
31
1019

2.1.3. Идентификаторы.

Любая последовательность букв и цифр, начинающаяся с буквы, называется идентификатором.

Примеры:

```
X
Y1
TEMP
A1Q25
```

Идентификаторы не имеют какого-либо навсегда присущего им внутреннего смысла. Они служат лишь для обозначения каких-либо объектов (в основном, значений или слогов), играя роль их «наименований». В частности, переменная представляется идентификатором. Имеется полная свобода в выборе идентификатора для обозначения того или иного объекта. Однако, в одной программе каждый идентификатор может использоваться для обозначения только одного какого-либо объекта.

Для наименования значений (переменных) обычно используются отдельные буквы, являющиеся частным случаем идентификатора. Однако из-за ограниченности их числа часто приходится вводить буквы с индексами, много удобств имеется в мнемоническом обозначении переменных, отражающем смысловую сторону значений, например:

```
TIME
MAX
```

Удобно использовать в программе и латинские названия греческих букв, фигурирующих в постановке задачи, например:

ALPHA
BETA

Все эти возможности и много других дает идентификатор.

Несмотря на то, что длина идентификатора может быть произвольна (лишь бы он размещался в одной строчке записи), различение идентификаторов в данной системе производится по четырем первым знакам. Идентификаторы, у которых совпадают эти четыре первых знака, считаются представителями одного и того же идентификатора.

Так, идентификаторы: BETA 1 и BETA 2 не различимы.

Каждый идентификатор, используемый в программе, должен быть описан (см. ниже).

2.1.4. Подчеркнутые слова.

Любая последовательность допустимых знаков (как правило, букв и цифр), подчеркнутая снизу, называется подчеркнутым словом. При начертании на бумаге подчеркнутого слова «подчеркивание» имеет обычный графический смысл. При перфорации же под-

черкнутых слов перед каждым знаком, входящим в подчеркиваемую последовательность, будет перфорироваться специальный символ подчеркивания. Примеры:

SIN
GO TO
1 FLOAT

В языке имеется заранее фиксированный набор подчеркнутых слов и каждое из них играет вполне определенную роль. Подчеркнутые слова используются для обозначения операций или в качестве служебных символов.

Хотя набор подчеркнутых слов, используемых для обозначения операций, заранее фиксирован, он легко может быть изменен при модификации языка. Поэтому полный перечень их мы здесь приводить не будем, отсылая к таблице операций (см. п.3). Следующие подчеркнутые слова используются в качестве служебных символов:

ARRAY
BOOLEAN
END
FALSE
INTEGER
REAL
TRUE

Назначение служебных символов будет ясно из дальнейшего изложения.

Подчеркнутые слова определяются своими первыми четырьмя подчеркнутыми знаками. Поэтому при написании программы не обязательно их каждый раз выписывать полностью, можно делать любые сокращения с обязательным сохранением лишь первых четырех подчеркнутых знаков.

2.2. Символические инструкции.

Узловым понятием языка является символическая инструкция. Символическая инструкция занимает в записи отдельную строчку и состоит в общем случае из трех частей: левого описания, символического слога и правого описания. Основу каждой символической инструкции составляет символический слог, который в ней всегда содержится. Левое иди правое описание или оба вместе могут в символической инструкции отсутствовать.

2.2.1. Символические слоги.

Символический слог является прообразом слога на внутреннем языке и поэтому может представлять либо операции, либо значение, либо ссылку.

2.2.1.1. Слоги — операции.

Слоги — операции в СИМПОЛИЗе 64 задаются либо знаками:

+ — x / =

либо подчеркнутыми словами, описанными в таблице операций (см.п.3).

Примеры:

+

INDEX

=

В символической инструкции, содержащей слог-операцию, всегда отсутствует правое описание.

Выполнение слога-операции приводит к выполнению соответствующей подпрограммы.

2.2.1.2. Слоги — значения.

В СИМПОЛИЗе 64 слог-значение может быть либо целым без знака, либо булевым значением, либо адресом.

Целое без знака определено в п. 2.1.2.

Булево значение представляется одним из двух подчеркнутых слов:

TRUE («истина»)
FALSE («ложь»)

Адрес в СИМПОЛИЗе 64 представляется идентификатором, перед которым расположена закрывающая круглая скобка и за которым расположена открывающая круглая скобка.

Примеры слогов-значений.

10
FALSE
)MAX(

В символической инструкции, содержащей слог-значение первых двух типов, всегда отсутствует правое описание.

Выполнение такого слога приводит к занесению в магазин, соответствующего значения. Для первых двух типов слогов эти значения однозначно определяются самим символическим слогом.

Третий же тип слога порождает некоторый обобщенный адрес, который на внутреннем языке заменяет данный идентификатор (см. п.1.3.4). Конкретное его значение зависит, главным образом, от месторасположения описания этого идентификатора. Но при составлении программы существенно не конкретное значение этого обобщенного адреса, а его смысл — то, что он является наименованием (адресом) определенного объек-

та, а это однозначно определяется идентификатором (согласно его описанию), который представляется этим обобщенным адресом.

Конкретные значения обобщенных адресов, заменяющих идентификаторы программы, выдаются на печать автокодиром только после получения программ в ИНПОЛИЗе и могут быть использованы, например, для контроля за работой этой программы.

2.2.1.3. Слоги — ссылки.

Слоги-ссылки изображаются просто идентификатором.

Примеры:

X1
RHO

Выполнение такого слога сводится к подстановке вместо этого слога того объекта, который обозначается данным идентификатором, и выполнении его как нового слога. Если этот объект был значением, то в результате всего этого в магазин будет занесено это значение (в данном случае, может быть, и вещественное). Поэтому (так как отмеченный случай является основным и практически достаточным для составления программ) можно сказать, что выполнение такого слога сводится, за исключением некоторых специальных случаев, к занесению в магазин значения, которое

обозначается данным идентификатором. Остальные случаи вытекают из первого определения. В п.2.2.3 будет дан пример более «хитрого» использования ссылок.

2.2.2. Описания.

Каждое описание вводит обозначение некоторого объекта или, другими словами, каждое описание придает соответствующему идентификатору конкретный смысл. Именно при обработке описаний идентификаторам ставятся в соответствие конкретные обобщенные адреса, тем самым производится размещение в памяти используемых объектов.

Каждый идентификатор должен быть описан. Эти описания могут быть либо левыми, либо правыми.

2.2.2.1. Левые описания.

Левое описание всегда является описанием метки. Меткой здесь называется идентификатор, которым обозначается некоторый слог программы.

Описание метки расположено всегда непосредственно перед тем слогом (символическим), который обозначается описываемым идентификатором, и состоит из этого идентификатора и следующего за ним многоточия. Под многоточием понимается две или более расположенных друг за другом точек.

Примеры описаний меток:

L1…
NEXT..
BEGIN

Рекомендуется для большей наглядности после описываемого идентификатора всегда ставить троеточие.

Так как все слоги программы считываются различными объектами независимо от того, совпадают ли они по написанию или нет, месторасположение описания каждой метки определяется однозначно (в отличие от месторасположений других описаний, см. п. 2.2.2.2).

В программе метки используются главным образом для организации разветвлений вычислительного процесса (условных и безусловных переходов, см. описание операций GO TO и ELSE в п. 3). Однако их можно использовать и для наименования значений (переменных), которые помечаются вместо соответствующего слога (например, расположенного в конце программы). Однако эти значения могут быть либо целыми, либо булевыми, так как каждый слог представляется коротким словом.

Вообще говоря, метки позволяют организовать и некоторую перестройку программы в процессе её выполнения (например, производить изменение слогов-

значений), однако в этом нет большой необходимости. Следует, кроме того, подчеркнуть, что в СИМПОЛИЗе 64 только использование метки в качестве ссылки позволяет при выполнении программы подставлять вместо неё не значение, а произвольный слог, что лежит в основе некоторых «хитрых» приемов программирования.

2.2.2.2. Правые описания.

Правые описания бывают двух видов: описание типов и описания массивов. Каждое правое описание располагается вслед за символическим слогам, содержащим описываемый идентификатор (в качестве ссылки или в качестве значения). При обработке автокодиром такого описания производится равмещение в памяти соответствующего объекта.

Каждый идентификатор может многократно употребляться в программе, и его описание может быть помещено после любого символического слога, содержащего этот идентификатор, но только первое такое описание каждого идентификатора принимается во внимание. Последнее связано с тем, что каждый идентификатор в одной программе может использоваться для обозначения только одного какого-нибудь объекта, но есть ряд удобств в допущении повторного описания этого идентификатора, идентичного первоначальному (обработке этого повторного описания привела бы к

повторному размещению в памяти этого объекта, что эквивалентно введению нового объекта).

Таким образом, порядок размещения объектов в памяти машины целиком определяется порядком расположения в программе описаний идентификаторов.

2.2.2.2.1. Описания типа.

Допустимы следующие три описания типа:

REAL
INTEGER
BOOLEAN

Первое описание резервирует в памяти место для длинного слова, два последних описания — для короткого слова. При этом описываемому идентификатору будет ставится в соответствие обобщенный адрес такого слова.

В СИМПОЛИЗе 64 не накладывается каких-либо ограничений на типы значений, которые будут представлять эти слова. В частности, переменной (идентификатору), описанной как REAL можно присваивать не только вещественные, но и целые или булевы значения (только в этом случае часть места в памяти будет использоваться нерационально), а также и другие типы значений, для которых нет специальных описаний (например, для строк, которые используются

лишь в операциях вводе и выводе). Однако в этих случаях нужно потом правильно использовать эти значения (применять к ним операции, соответствующие их типу, а не типу, который приписан соответствующему идентификатору). В этом смысле излишне наличие в языке двух описаний:

INTEGER и BOOLEAN

можно было бы обойтись одним из них, но их наличие все же позволяет сделать программу более наглядной. Следует обратить внимание на то, что короткие слова могут представлять вещественные значения, примерно, с полутора верными десятичными знаками, поэтому не рекомендуется вещественные значения помещать на место коротких слов.

2.2.2.2.2. Описания массива.

В СИМПОЛИЗе 64 идентификатор может обозначать целую группу значений одного типа, называемую массивом. Для ссылки на какое-либо значение из этого массива (на компоненту массива) требуется указать наименование этого массива и порядковый номер этой компоненты в данном массиве, причем компоненты в каждом массиве нумеруются, начиная с нуля.

Описание массива состоит из служебного подчеркнутого слова ARRAY, за которым следует целое

без знака, а в конце располагается указатель типа значений в этом массиве, одно из трех подчеркнутых слов:

REAL
INTEGER
BOOLEAN

Примеры:

ARRAY 135 REAL
ARRAY 9 INTEGER
ARRAY 3 BOOLEAN

Целое без знака, указанное в описании массива, определяет число компонент в этом массиве. При обработке описания массива в памяти резервируется место именно для такого числа длинных или коротких слов.

Ссылку на нулевую компоненту массива можно производить двояко: по общему правилу (сформулированному выше) и с помощью только идентификатора этого массива так, как будто этот идентификатор обозначает только эту нулевую компоненту (см. п. 2.2.2.2.1). Последний способ более компактен.

2.2.3 Примеры символических инструкций.

```
M1… 15
      )EPS(
LOS.. STOP
      )A(ARRAY 23 REAL
SS… N INTEGER
      +
```

Теперь приведем пример более «хитрого» использования меток. Если в программе имеются символические инструкции:

```
L… +
M… L,
```

то инструкции:

```
L
M
```

по своему действию эквивалентны друг другу и третьей инструкции:

```
+
```

2.3. Программа

Программа в СИМПОЛИЗе 64 представляет собой последовательность символических инструкций, которая замыкается символической инструкцией:

END

и не содержит этой инструкции внутри себя. Инструкция END имеет единственное назначение — ограничивать программу, она не оказывает никакого влияния на выполнение программы.

При кодировании программы на внутреннем языке каждая символическая инструкция, кроме END, переходит в определенный слог ИНПОЛИЗа, т.е. при кодировании устанавливается взаимнооднозначное соответствие между символическими инструкциями : СИМПОЛИЗа 64 и слогами ИНПОЛИЗа.

2.3.1. Примеры простейших программ.

В нижеприводимых примерах используются операции, описанные в п.3. При первом чтении эти примеры можно рассматривать как чисто формальную иллюстрацию понятия программы. Однако после ознакомления с п.3 к этим примерам целесообразно вернуться с целью их содержательного разбора.

Печать текста.

```
)TEXT(ARRAY 25 REAL
S READ
)TEXT(
S PRINT
STOP
END
```

Отыскание на перфоленте числа 5.

```
BEGIN…) I(INTEGER
1
I READ
I
5
I EQUAL
)BEGIN(
ELSE
STOP
END
```

Решение уравнений вида $A_0 x + A_1 = 0$:

```
G…) A(ARRAY 2 REAL
2
R READ
)STRING(REAL
```

```
S READ
)X(
)A(
1
COMPONENT
A
/
R NEG
=
)STRING(
X REAL
2
3
R PRINT
)G(
GO TO
END
```

3. Таблица операций.

Для описания операций удобнее рассматривать магазин с несколько иной точки зрения. Дело в том, что все операции совершаются над содержимым ближайших к окну ячеек магазина. Поэтому целесообразно вести нумерацию ячеек магазина относительно окна, считая окно за нулевую ячейку. Но, как известно, окно магазина от операции к операции может переме-

щаться, поэтому от операции к операции может меняться и указанная выше нумерация ячеек.

Для лучшего понимания этих обозначений представим себе мысленно, что есть сетка с семью клетками, которая может быть наложена на последовательность ячеек памяти, представляющую собой магазин. Клетки занумерованы: нулевая, первая, и так далее, всего клеток семь. Наложение происходит таким образом, что каждая клетка покрывает одну и только одну ячейку памяти и нулевая клетка совпадает с окном магазина.

После наложения сетки на магазин в каждой клетке «видна» некоторая величина, которая записана в соответствующей ячейке магазина. Будем считать, что в нулевой клетке видна величина S_0, в первой S_1 и т.д.

Нулевая клетка всегда наложена на одну из не занятых ячеек магазина. Остальные клетки покрывают магазин только по мере записи в магазин информации.

При каждой записи, покрытие магазина сеткой увеличивается еще на одну клетку, при этом вся сетка сдвигается.

Сдвиг сетки происходит следующим образом: нулевая клетка покрывает еще не занятую ячейку, первая клетка — ячейку, покрытую раннее нулевой, вторая — покрытую раннее первой и т.д.

При определении каждой операции в таблице указано, какие величины являются операндами для

данной операции. После выполнения операции, операнды, участвовавшие в операции исключаются из магазина. Результат, как правило, записывается в магазин. При этом, для одноместных операций сдвига сетки не происходит, для двухместных происходит сдвиг сетки на одну клетку. Направление сдвига в этом случае противоположно направлению сдвига при записи, то есть первая клетка будет покрывать ячейку, в которой записан результат, вторая – ячейку, которая раньше была покрыта третьей и т.д. Соответственно, со сдвигом сетки изменится и название величин, записанных в магазине, S_2 на S_1, S_3 на S_2 и т.д.

Общее число ячеек, покрытых сеткой после выполнения двухместной операции, результат которой записывается в магазин, уменьшается на единицу, иными словами, число занятых ячеек магазина на единицу сокращается. Если результат операции не записывается в магазин, сетка сдвигается на столько ячеек, сколько операндов участвовало в операции.

Сдвиг окна магазина и соответствующее изменение числа занятых ячеек магазина (сдвиг сетки) в таблице операций обозначается вертикальной стрелкой ↓. Слева от стрелки стоит величина S_i; справа – S_j : S_i ↓ S_j. Разность $i-j$ указывает величину и направление сдвига. Если $i-j>0$, то это означает, что освободилось $i-j$ ячеек магазина (и соответственно переместилось его окно). Если $i-j<0$ это означает, что после выполнения данной операции число занятых

ячеек магазина увеличилось на $|i-j|$ (и соответственно переместилось окно). Например, обозначение S2 ↓ S1 надо понимать так, что первая клетка теперь будет покрывать величину, которую раньше покрывала вторая, аналогично сдвинутся и клетки, покрывающие все остальные ячейки, а общее число занятых ячеек магазина уменьшится на единицу.

3.1. Арифметические операции.

Арифметические операции можно разделить на два класса: 1) арифметические операции над вещественными значениями, 2) арифметические операции над целыми значениями.

Символы +, -, x, / означают название соответствующих операций над вещественными значениями. Для остальных операций в первом случае перед названием ставится буква R, во втором — I. Например, R ABS, I ABS.

3.1.1. Операции над вещественными значениями.

Для рассматриваемого класса операций результат является, как правило, вещественным, однако, две операции составляют исключение. Поэтому данный класс операций разбит на два подкласса, характеризующиеся типом результата, получаемого в ре-

зультате выполнения операции из соответствующего подкласса.

3.1.1.1. Операции с вещественным результатом.

№ № п/п	Название операции	Содержание
1.	+	$S_2+S_1\Rightarrow S_2\downarrow S_1$
2.	-	$S_2-S_1\Rightarrow S_2\downarrow S_1$
3.	x	$S_2\times S_1\Rightarrow S_2\downarrow S_1$
4.	/	$S_2/S_1\Rightarrow S_2\downarrow S_1$
5.	R NEG	$-S_1\Rightarrow S_1$
6.	R ABS	$\lvert S_1\rvert\Rightarrow S_1$
7.	INVERSE	$1/S_1\Rightarrow S_1$

3.1.1.2. Операции с целым результатом.

№ № п/п	Название операции	Содержание
8.	R SIGN	$sign(S_1)\Rightarrow S_1$
9.	ENTIER	$entier(S_1)\Rightarrow S_1$

Примечание:

$$sign(x)=\begin{cases}1, при\ x>0\\0, при\ x=0\\-1, при\ x<0\end{cases}$$

entier(x) – есть наибольшее целое, не превосходящее x.

3.1.2. Операции над целыми значениями.

Для операций этого класса результат есть также целое число, за исключением операции деления, результат которой вещественный. Кроме того, в операционной системе предусмотрена возможность с помощью операций 1 FLOAT и 2 FLOAT переходить от целых значений к значениям вещественным. Поэтому в данном классе арифметических операций выделены два подкласса.

3.1.2.1. Операции с целым результатом.

№ № п/п	Название операции	Содержание
10	I PLUS	$S_2+S_1\Rightarrow S_2\downarrow S_1$
11	I MIN	$S_2-S_1\Rightarrow S_2\downarrow S_1$
12	I MUL	$S_2\times S_1\Rightarrow S_2\downarrow S_1$

№ № п/п	Название операции	Содержание
13	I NEG	$-S_1 \Rightarrow S_1$
14	I ABS	$\lvert S_1 \rvert \Rightarrow S_1$
15	I SIGN	$sign(S_1) \Rightarrow S_1$

3.1.2.2. Операции с вещественным результатом.

№ № п/п	Название операции	Содержание
16	I DIV	$S_2 / S_1 \Rightarrow S_2 \downarrow S_1$
17	1 FLOAT	$float(S_1) \Rightarrow S_1$
18	2 FLOAT	$float(S_2) \Rightarrow S_2$

Примечание: float(x) есть вещественное значение, совпадающее с целым значением X.

3.1.3. Операции над значениями разных типов.

В описываемой системе среди арифметических операций есть только одна операция, один из операндов которой представляет вещественное значение, а другой целое. Эта операция возведения в степень, при этом показатель степени целое число, а основание вещественное, результат также вещественный.

№ № п/п	Название операции	Содержание
19.	POWER	$S_2^{S_1} \Rightarrow S_2 \downarrow S_1$

3.2. Операции отношения.

Операции отношения определяют истинность некоторых высказываний относительно двух значений. Необходимость в этом появляется при разветвлении вычислительного процесса. Высказывание может быть истинным или ложным, поэтому результат операции является всегда булевым. Если отношение S2*S1 имеет место (где * может быть: >, <, =, ≥, ≤, ≠), то результат есть TRUE, если данное отношение не имеет место, то результат – FALSE.

Операции отношения определены для вещественных и для целых значений.

3.2.1. Операции над вещественными значениями.

№ № п/п	Название операции	Содержание
20.	R MORE	$S_2 > S_1 \Rightarrow S_2 \downarrow S_1$
21.	R LESS	$S_2 < S_1 \Rightarrow S_2 \downarrow S_1$
22.	R EQUAL	$S_2 = S_1 \Rightarrow S_2 \downarrow S_1$

№ № п/п	Название операции	Содержание
23.	R NLESS	$S_2 \geq S_1 \Rightarrow S_2 \downarrow S_1$
24.	R NMORE	$S_2 \leq S_1 \Rightarrow S_2 \downarrow S_1$
25.	R NEQUAL	$S_2 \neq S_1 \Rightarrow S_2 \downarrow S_1$

3.2.2. Операции над целыми значениями.

№ № п/п	Название операции	Содержание
26.	I MORE	$S_2 > S_1 \Rightarrow S_2 \downarrow S_1$
27.	I LESS	$S_2 < S_1 \Rightarrow S_2 \downarrow S_1$
28.	I EQUAL	$S_2 = S_1 \Rightarrow S_2 \downarrow S_1$
29.	I NLESS	$S_2 \geq S_1 \Rightarrow S_2 \downarrow S_1$
30.	I NMORE	$S_2 \leq S_1 \Rightarrow S_2 \downarrow S_1$
31.	I NEQUAL	$S_2 \neq S_1 \Rightarrow S_2 \downarrow S_1$

3.3. Логические операции.

Использование логических операций предполагает знакомство с элементами математической логики.

Логические операции определены только для булевых значений. Результат логических операций всегда является булевым.

№ № п/п	Название операции	Содержание	Примечание
32.	NOT	$\neg S_1 \Rightarrow S_1$	логическое отрицание
33.	OR	$S_2 \vee S_1 \Rightarrow S_2 \downarrow S_1$	логическое сложение
34.	AND	$S_2 \wedge S_1 \Rightarrow S_2 \downarrow S_1$	логическое умножение
35.	IDENT	$S_2 \equiv S_1 \Rightarrow S_2 \downarrow S_1$	установление эквивалентности

3.4. Операции следования.

Операции следования изменяют естественный порядок выполнения слогов программы. К этим операциям относятся безусловный и условный переходы и операция останова. При выполнении операций перехода адрес (наименование) слога, подлежащего выполнению, определяется согласно его метке (см. п.2.2.2.1), заданной в магазине в виде значения (адреса) величиной S_1. Условный переход предполагает, кроме того, первоначальное получение в магазине булева

значения, являющегося, например, результатом проверкинекоторого условия (выполнения операции отношения). Это значение представляется величиной S_2. Переход по метке в этом случае происходит только в том случае, если S_2 есть FALSE. В противном случае выполняется как обычно следующий по порядку слог.

Адрес очередного выполняемого слога будем обозначать буквой N. При описании всех операций предполагалось, что после выборки очередного слога и до его выполнения выполняется дополнительное действие:

$$N+1 \Rightarrow N$$

(подготовляющее переход к следующему слогу).

№ № п/п	Название операции	Содержание
36.	GO TO	$S_1 \Rightarrow N\,;\, S_2 \downarrow S_0$
37.	ELSE	если $\neg S_2$ то $S_1 \Rightarrow N\,;\, S_2 \downarrow S_0$
38.	STOP	Останов. Выполнение программы прекращается, N (в виде обобщенного адреса) посылается в регистр S машины «Сетунь». Работу можно продолжить нажатием кнопки «пуск» на пульте управления машины.

Пример*. Послать в магазин наибольшее из двух вещественных значений X и Y.

```
        X REAL
        Y REAL
    R MORE
    )M1(
    ELSE
    X
    )M2(
    GO TO
M1…Y
M2…
```

3.5. Операции присваивания.

Операция предназначена для записи информации из магазина в ячейку основной памяти.

№ № п/п	Название операции	Содержание	Примечание
39.	=	$S_1 \Rightarrow (S_2);$ $S_2 \downarrow S_0$	S_1 посылается в ячейку памяти с адресом S_2

*В дальнейшем будем предполагать, что приводимые здесь последовательности слогов являются частями каких-либо программ.

3.6. Операции доступа к массиву.

Действия с компонентами массивов могут быть двух типов:

1) засылка в магазин некоторой компоненты массива,

2) формирование адреса некоторой компоненты массива.

Одним из операндов обеих операций является адрес, соответствующий наименованию некоторого массива. Это значение представлено величиной S_2. Вторым операндом является номер компоненты (целое значение), представленный величиной S_1.

№ № п/п	Название операции	Содержание
40.	COMPONENT	$S_2+S_1\Rightarrow S_2;(S_2)\Rightarrow S_2\downarrow S_1$
41.	INDEX	$S_2+S_1\Rightarrow S_2\downarrow S_1$

Примеры. Вычислить $Y_1=X_1+A$, где Y и X – вещественные массивы, A – вещественное значение, I – целое значение.

```
)Y(
I
INDEX
)X(
```

I

COMPONENT

A

+

=

Вычислить $Z_1 = Z_0 + A$, где Z– вещественный массив, A – вещественное значение (приводим два варианта программы, см. левый и правый столбцы):

)Z(	)Z(
1	1
INDEX	INDEX
)Z(	Z
0	A
COMPONENT	-
-	=
=	

3.7. Операции дублирования и перестановки.

Данные операции могут быть полезны для сокращения или упрощения программы.

№ № п/п	Название операции	Содержание
42.	DUPL	$S_1 \Rightarrow S_0 \downarrow S_1$
43.	TRANSFER	$S_1 \Rightarrow a\,;\ S_1 \Rightarrow S_2\,;\ a \Rightarrow S_1\,;$ где а – некоторая промежуточная величина

3.8. Операции ввода и вывода*.

Операции ввода и вывода информации существенно используют удобства алфавитно-цифровых устройств ввода и вывода, позволяющие задавать информацию, а также получать результаты в удобной для человека форме. Допускается ввод и вывод текста. Числа и текст вводятся с любого из двух имеющихся фотовводов. Вывод производится как на перфорирующее устройство, так и на печатающую машинку. Ниже перечислены все операции ввода и вывода, включенные в операционную систему ПОЛИЗ 64.

R READ
I READ
S READ
} операция ввода

R PRINT
I PRINT
S PRINT
} операция вывода

*Данный пункт написан Бондаренко Н.В.

IN 1
IN 2
PUNCH
TIPE
} служебные операции

Здесь первые выделенные буквы R, I или S в названии операций (кроме служебных)обозначают тип информации, к которой применяется соответствующая операция (от слов REAL, INTEGER или STRING — строка). Представление чисел и текста при выводе следует тем же правилам, что и при вводе информации, поэтому возможно, например, ввести с помощью операции I READ числа, которые ранее были выведены с использованием операции I PRINT.

3.8.1. Представление алфавитно-цифровой информации.

Набор символов, воспринимаемых машиной (см. п.2.1.1.), включает в себя цифры, заглавные латинские буквы (буква O совмещена с нулем), знаки арифметических операций (+, -, ×, /, =), десятичную точку (·), открывающую и закрывающую круглые скобки, пробел (_) и некоторые управляющие символы: возврат каретки (вк), вызывающий печать с новой строки, символ «стоп» (ϙ), воспринимаемый машиной как сигнал окончания ввода или вывода информации, «подчеркивание» (пч) и, наконец, символы «переклю-

чения» регистров (цр — цифровой регистр и бр — буквенный регистр). Последние два символа требуют дополнительных пояснений. Дело в том, что большинство из указанных символов разбиты на пары, в каждую из которых входят символы, различающиеся только при печати — на перфоленте и в машине они представлены одной и той же комбинацией. Какой именно символ из пары представляет эта комбинация зависит от того, какой регистр (буквенный или цифровой) был последний раз «включен», т.е. какой символ из всех символов цр и бр в последовательности комбинаций перед данной комбинацией является самым правым.

Таким образом, для задания алфавитно-цифровой информации необходимо знать распределение символов по регистрам и тщательно следить за своевременным переключением регистров. Несвоевременное включение нужного регистра приводит к существенному искажению текста. Во избежание ошибок в этом смысле можно каждый символ задавать парой комбинаций, первая из которых есть символ переключения регистра. Однако многие из комбинаций ц р и б р будут в этом случае излишними. Ниже приводится распределение пар символов по регистрам.

Цифровой регистр	Буквенный регистр	Цифровой регистр	Буквенный регистр	Цифровой регистр	Буквенный регистр
6	A	V	J	4	T
7	B	W	K	5	U
8	C	X	L	)	(
9	D	Y	M	x	=
_	E	Z	N	пч	пч
-	F	0	P	бр	бр
/	G	1	Q	цр	цр
.	H	2	R	вк	вк
+	I	3	S	⚲	⚲

Пример 1. Запись слова CODE:

бр C цр O бр D бр E⚲

или, опустив последний (излишний) символ переключения регистров:

бр C цр O бр D E⚲

Пример 2. Запись чисел 1/100, -2.1416, +5 и 0.0014:

цр 0. 01 _ _ -2. 1416 _ + 5 вк
- 0. 0014⚲

3.8.2. Правила записи чисел.

На внешних носителях информации числа представляются в любой десятичной форме с фиксированной запятой, т.е. в виде целых чисел со знаком или без знака или в виде десятичных смешанных дробей со знаком или без знака.

Примеры записи чисел:

```
+10
5
-5.00
-1000.31
0.0008
+.8
-4
```

Числа, вводимые с помощью операции I READ должны быть представлены только в виде целых чисел со знаком или без знака.

На форму представления чисел при выводе накладываются дополнительные ограничения:

1. Положительный знак числа заменяется пробелом (_)

2. При использовании операции R PRINT десятичная точка выводится даже в случае отсутствия дробной части.

3. Вместо незначащих нулей целой части числа выводятся пробелы в соответствии с указанным при выполнении операции вывода форматом числа.

Числа на внешних носителях, предназначенные для ввода, отделяются друг от друга символами _ или вк; переключение регистров внутри вводимой числовой информации не допускается.

Вводимая информация должна быть разбита на группы (зоны) не более, чем в 150 символов. Практически удобно помещать в такую группу по 10 чисел (или меньше). В конце каждой такой группы должны стоять три символа ♀ и затем пропуск на перфоленте, примерно, в 15 см. При выводе между числами может быть помещен, вообще говоря, любой текст, но если выводимая информация предназначена для повторного ввода, -должны быть выполнены вышеуказанные требования.

3.8.3. Текстовая строка.

Под текстовой строкой будет понимать произвольную последовательность символов (включая и символы цр и бр), оканчивающуюся символом ♀ и не превышающую 150 символов.

Если текстовая строка содержит не более 6 символов, то такую строку будем называть короткой. В противном случае строку будем называть длинной.

Каждая строка в СИМПОЛИЗЕ 64 обозначается идентификатором, который описывается типом REAL для короткой строки и массивом REAL для длинной строки. Для резервирования в памяти достаточного количества ячеек под длинную строку нужно иметь в виду, что на место каждой вещественной компоненты массива может быть помещено ровно шесть символов. Таким образом массив отведенный для длинной строки требует не более 25 длинных ячеек памяти.

В программе может быть только один идентификатор длинной строки и его описание в программе должно предшествовать всем правым описаниям.

Над строками никаких операций, кроме ввода и вывода, не производится, причем вывод строки означает просто дублирование ранее введенной строки устройством вывода.

Строка никогда не посылается в магазин. Для указания какой-либо строки в качестве операнда в магазин посылается ее адрес, соответствующий идентификатору, которым обозначена эта строка.

Строка, используемая в качестве операнда для операций R PRINT или I PRINT, называется ведущей строкой. Она выводится непосредственно перед числом и служит для управления форматом таблиц, которые строятся из выводимых чисел. Она позволяет отпечатать выводимое число с новой строки или в той же строке, что и предыдущее число, отступив от него на

некоторое количество пробелов. Ведущая строка, как правило, является короткой.

Текстовые строки используются для вывода на печать заголовка задачи (с помощью операции S PRINT) или в качестве заголовка страницы некоторой таблицы и т.п. Но во всех случаях она должна быть первоначально введена в машину с помощью операции S READ.

Пример 1. Пусть требуется напечатать таблицу вида:

X	Y	F(X)	G(Y)
0.1	-0.11	-4.1418	0.00163
0.2	-0.13	-3.8992	0.02938
……			

Тогда строка:

«цр _ _ _ _ _ вк
_ _ _ _ _ вк
_ _ X _ _ _ _ _ _ Y _ _ _ _ _ _ брF (црX) _ _
_ _ _ _ брG (црY) ⸮»

может быть использована в качестве заголовка страницы указанной таблицы, строка «вк ⸮» — в качестве ведущей строки для чисел первоначально введена в машину с первого столбца, строка «_ _ _ ⸮»в каче-

стве ведущей строки для чисел второго, третьего и четвертого столбцов. Наличие символов переключения регистров в ведущих строках здесь необязательно, так как они начинают выводиться здесь всегда при включенном цифровом регистре.

Пример 2. Требуется вывести на перфорирующее устройство n вещественных чисел. Ведущей строкой для каждого из них может быть любая из строк:

«вк ⚲»
«вк _ ⚲»
«цр _ ⚲»

3.8.4. Служебные операции.

Служебные операции осуществляют настройку операций ввода-вывода на любое из имеющихся устройств (два фотоввода, выводной перфоратор, печатающая машинка). В дальнейшем будем называть это действие включением устройства, а само устройство включенным.

Фактически, служебные операции лишь изменяют некоторые машинные слова в соответствующих программах операционной системы ПОЛИЗ 64.

Четырем имеющимся устройствам ввода-вывода соответствуют служебные операции:

№ №	Название	Содержание

п/п	операции	
44.	IN 1	Включение фотоввода №1
45.	IN 2	Включение фотоввода №2
46.	PUNCH	Включение перфоратора
47.	TIPE	Включение печатающей машинки.

Включение фотоввода №1 (№2) означает, что в дальнейшем вплоть до следующего выполнения служебной операции, соответствующей устройствам ввода, алфавитно-цифровая информация будет вводиться только с фотоввода №1 (№ 2). Включение перфоратора (печатающей машинки) означает, что с этого момента вплоть до следующего выполнения соответствующей служебной операции вывод информации будет производиться только на перфоратор (на печатающую машинку).

Любая из перечисленных операций не изменяет содержимого магазина, поэтому ее можно писать в любом месте программы до соответствующей операции ввода или вывода.

Следует иметь в виду, что до выполнения первой служебной операции включенными будут фотоввод №2 и печатающая машинка. Поэтому, если предполагается использование только этих устройств ввода и вывода, употребление служебных операций излишне.

3.8.5. Операции ввода.

Операции ввода осуществляют ввод* массива числовых данных (вещественных или целых, представленных как указано в п. 3.8.2) в память машины, а также ввод текстовой строки. Операндом каждой из этих операций является адрес, показывающий куда нужно ввести информацию. Этот адрес соответствует идентификатору, описанному как массив или как тип. Для операций ввода числовой информации имеется также и другой операнд, представляющий целое значение, равное числу вводимых компонент. При этом, если адрес соответствует идентификатору, описанному как некоторый тип, то это число должно быть равно 1. Операция ввода булевских значений в данной системе отсутствует. В случае необходимости булевские значения можно вводить в виде целых чисел, равных 0 или 1*.

№ № п/п	Название операции	Содержание
48.	R READ	$R\,READ(S_2, S_1);\ S_2 \downarrow S_0$
49.	I READ	$I\,READ(S_2, S_1);\ S_2 \downarrow S_0$
50.	S READ	$S\,READ(S_1);\ S_2 \downarrow S_0$

*Ввод числовых данных включает в себя и перевод в троичную систему с формой записи, соответствующей указанному типу.
*Однако, на языке ИНПОЛИЗ представление целого значения 1 не совпадает с представлением значения TRUE (см.п.1.3)

Примечания. Здесь R READ(S_2,S_1) обозначает процедуру ввода массива вещественных значений с включенного устройства ввода и запись его по адресу S_2 (S_1 — целое значение, равное числу компонент массива); I READ(S_2,S_1) обозначает аналогичную процедуру для целых значений; S READ(S_1) обозначает процедуру ввода с включенного устройства текстовой строки по адресу S_1.

3.8.6. Операции вывода.

При выводе числовой информации одной из главных задач является обеспечение печати чисел в нужном формате и организация размещения чисел на листе бумаги.

Формат вещественного числа задается количеством позиций m, до десятичной точки и количеством десятичных знаков r дробной части числа. Формат целого числа определяется количеством позиций l, отведенных под число. Причем m и l должны быть не меньше количества значащих цифр целой части числа. В противном случае предусмотрена печать слова, сигнализирующего о неправильном задании формата. Целые положительные значения m и r или l являются операндами соответствующих операций вывода чисел R PRINT или I PRINT.

Управление размещением чисел на листе бумаги (печать пробелов, переход к новой строке и т.п.) осуществляется с помощью ведущей строки (см. п. 3.8.3), которая выводится непосредственно перед числом. Адрес ведущей строки и само число также являются операндами операций вывода числовой информации.

Примечание. Формат вещественного числа будет включать m+r+2 позиции (к цифровым позициям добавляются позиции знака и десятичной точки). Формат целого числа будет включать l+1 позицию (к цифровым позициям добавляется позиция знака).

Операция вывода текстовой строки S PRINT имеет единственный операнд — адрес соответствующей выводимой строки. Вывод булевых значений (TRUE и FALSE) может осуществляться также с помощью операции S PRINT.

№ № п/п	Название операции	Содержание
51.	R READ	$R\,READ(S_4, S_3, S_2, S_1);\ S_4 \downarrow S_0$
52.	I READ	$I\,READ(S_3, S_2, S_1);\ S_3 \downarrow S_0$
53.	S READ	$S\,READ(S_1);\ S_1 \downarrow S_0$

Примечание. Здесь R PRINT (S_4, S_3, S_2, S_1) представляет процедуру вывода в десятичной форме

вещественного значения S_3, формат которого задается целыми значениями S_2 и S_1, S_2=m, S_1=r. Перед числом выводится ведущая строка S_4.

I PRINT (S_3, S_2, S_1) представляет процедуру вывода в десятичной форме целого числа S_2, формат которого задается величиной S_1 (S_1=l). Перед числом выводится ведущая строка S_3. S PRINT(S_1) представляет процедуру вывода текстовой строки с адресом S_1.

3.8.7. Пример.

Приведем образец программы с употреблением операций ввода и вывода, решающей задачу примера 1 п. 3.8.3.

Пусть требуется вывести десять строк таблицы и F(X)=sin*X*, G(Y)=cos*Y*. Значения X и Y берутся с перфоленты.

Обозначим строку заголовка таблицы идентификатором STR1. После вывода заголовка на печать нет необходимости в дальнейшем хранении строки STR1 в памяти машины, поэтому на её место можно ввести любой массив чисел или другую текстовую строку. В нашем примере будет введена новая строка «вк ⚲». Текстовую строку «_ _ _ ⚲» обозначим идентификатором STR2. Далее, числа будем вводить с фотоввода №1, текст — с фотоввода №2.

При этих условиях программа на языке СИМПОЛИЗ 64 будет иметь следующий вид:

```
      )STR1( ARRAY 9 REAL
      S READ
      )STR1(
      S PRINT
      ) STR1 (
      S READ
      )STR2( REAL
      S READ
      IN 1
      )X( ARRAY 10 REAL
      10
      R READ
      )Y( ARRAY 10 REAL
      10
      R READ
      )I( INTEGER
      0
      =
M1…) STR 1(
      )X(
      I
      COMPONENT
      1
      1
      R PRINT
      )STR 2(
      )Y(
```

```
I
COMPONENT
1
2
R PRINT
)STR 2(
)X(
I
COMPONENT
SIN
1
4
R PRINT
)STR 2(
)Y(
I
COMPONENT
COS
1
5
R PRINT
)I(
I
1
I PLUS
=
I
10
```

```
        I EQUAL
        )M1(
        ELSE
M2…  STOP
        END
```

3.9. Операции вычисления элементарных функций.

Данные операции определены только для вещественных значений.

№ № п/п	Название операции	Содержание
54.	SQRT	$\sqrt{S_1} \Rightarrow S_1$
55.	SIN	$\sin(S_1) \Rightarrow S_1$
56.	COS	$\cos(S_1) \Rightarrow S_1$
57.	LN	$\ln(S_1) \Rightarrow S_1$
58.	EXP	$\exp(S_1) \Rightarrow S_1$

4. Примеры программы.

4.1 Постановка задачи.

Вычислить сумму:

$$\sum_{I=1}^{20} (X_0 \cdot \sin X_1 + Y_0 \cdot \cos Y_1)$$

и вывести ее на печать с двумя знаками перед запятой и с тремя знаками после запятой.

Числа X_i и Y_i, а также вся требующаяся дополнительная информация должна вводиться с перфоленты.

4.2 Программа в СИМРОЛИЗе 64.

```
)STRING( ARRAY 10 REAL
S READ
)X(
21
R READ
)Y( ARRAY 21 REAL
21
R READ
)I(
1
=
)STRING(
0
```

```
M1…)X( ARRAY 21 REAL
       0
       COMPONENT
       )X(
       I INTEGER
       COMP.
       SIN
       X
       Y
       )Y(
       I
       COMP.
       COS
       X
       +
M2… +
M3…)I(
       I
       1
       I PLUS
       =
       I
       21
       I EQUAL
       )M1(
       ELSE
       2
       3
```

```
      R PRINT
M4… STOP
      END
```

Литература

1. Е.А.Жоголев. Интерпретатор ПОЛИЗ-63. Ж. вычисл. матем. и матем.физ., 1965, 5, № I, 67-76.

2. Н.Б.Лебедева. Автопереводчик ПОЛИЗ. Отчет ВЦ МГУ, №35 АП 1964.

3. Е.А.Жоголев, Л.В.Есакова. Интерпретирующая система ИП-3. В данной серии, вып. 4, 1964.

4. Н.В.Бондаренко. Система программ ввода и вывода алфавитно-цифровой информации для ИП-3. В данной серии, вып.8, 1965.

5. Е.А.Жоголев. Система команд и интерпретирующая система для машины «Сетунь». Ж. вычисл. матем. и матем. физ., 1961, I, № 3, 499-512.

Издано в 1964 году:

Выпуск 1.

Жоголев Е.А. ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ И МАТЕМАТИЧЕСКОЕ ОБСЛУЖИВАНИЕ МАШИНЫ «СЕТУНЬ».

Выпуск 2.

Фурман Г.А. ИНТЕРПРЕТИРУЮЩАЯ СИСТЕМА ДЛЯ ДЕЙСТВИЙ С КОМПЛЕКСНЫМИ ЧИСЛАМИ (ИП-4).

Выпуск 3.

Франк Л.С, Рамиль Альварес Х. ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ЗНАЧЕНИЙ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ ДЛЯ ИП-2.

Выпуск 4.

Жоголев Е.А., Есакова Л.В. ИНТЕРПРЕТИРУЮЩАЯ СИСТЕМА ИП-3. Поправка к выпуску 4 опубликована в выпуске 9 (1965 г.)

Выпуск 5.

Фурман Г.А. ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ВСЕХ КОРНЕЙ МНОГОЧЛЕНА ДЛЯ ИП-4.

Выпуск 6.

Прохорова Г.В. ИНТЕРПРЕТИРУЮЩАЯ СИСТЕМА ДЛЯ ДЕЙСТВИЙ С ПОВЫШЕННОЙ ТОЧНОСТЬЮ (ИП-5).

Издано в 1965 году:

Выпуск 7.

Гордонова В.И. ТИПОВАЯ ПРОГРАММА РАСЧЕТА КОРРЕЛЯЦИОННЫХ И СПЕКТРАЛЬНЫХ ФУНКЦИЙ.

Выпуск 8.

Бондаренко Н.В. СИСТЕМА ПОДПРОГРАММ ВВОДА И ВЫВОДА АЛФАВИТНО-ЦИФРОВОЙ ИНФОРМАЦИИ ДЛЯ ИП-3.

Выпуск 9.

Черепенникова Ю.Н. НАБОР ПОДПРОГРАММ ДЛЯ ВВОДА И ВЫВОДА ЧИСЛОВОЙ ИНФОРМАЦИИ В СИСТЕМЕ ИП-2.