ВОПРОСЫ ТЕОРИИ И ПОСТРОЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ
(Вычислительные системы)
1979 год Выпуск 80

УДК 681.142.2; 681.31

ПРОГРАММИРУЮЩАЯ СИСТЕМА Р-ЛЯПАС. ИНСТРУМЕНТАЛЬНЫЙ ЯЗЫК

В.А.Воробьев

I. Общая характеристика ПС-Р-ЛЯПАС

Задача статьи - описание языка Р-ЛЯПАС, который является инструментальным языком программирующей системы (ПС) Р-ЛЯПАС, реализованной на ЭВМ БЭСМ-6.

I.I. Язык Р-ЛЯПАС предназначен для:

I) представления алгоритмов анализа и синтеза дискретных систем;

2) моделирования дискретных систем;

3) системного программирования.

Объединение этих трех целей, во-первых, возможно, так как большинство соответствующих задач имеет единую комбинаторно-логическую природу, и, во-вторых, необходимо, так как перечисленные проблемы имеют комплексный характер и требуют для своего решения на ЭВМ больших систем программ.

I.2. Основные требования, которым должна удовлетворять ПС: эффективное использование возможностей ЭВМ, преемственность программирования, сохранение простоты и наглядности программ и систем программ при значительном росте их объемов. Эти противоречивые требования с необходимостью приводят к построению о т к р ы т ы х, растущих систем и соответственно языков программирования. Конкретнее это означает:

I) многоуровневость инструментального языка, так что каждый последующий уровень опирается на предыдущий. Базовый уровень должен обеспечить удачный компромисс между машинной ориентированностью, дающей эффективность, и независимостью, обеспечивающей преемственность при переходе на новые ЭВМ;

2) иерархичность и модульность программирующей системы и всех продуктов деятельности пользователей. В идеале каждая программа должна состоять из модулей и сама быть модулем, каждому модулю должна сопоставляться макрооперация (макрос) инструментального языка, и с момента введения этого модуля в ПС соответствующий макрос входит в язык. Таким образом, использование открытой системы совпадает с ее наращиванием и обогащением языка. Совершенно аналогичные требования можно предъявить не только к набору операций языка, но и к набору операндов, их типов, структур, форматов и т.д.

I.3. Только что описанная идеалогия получила широкое развитие в программирующих системах на базе языка ЛЯПАС [I]. Непосредственным предшественником ПС-Р-ЛЯПАС были базовая моделирующая система Р-ЛЯПАС для ЭВМ "Минск-22" [2,3], ПС-ЛЯПАС-7I [4] и ПС-ЛЯПАС-70 для ЭВМ БЭСМ-6 [5].

В отличие от недавно разработанной ПС-ЛЯПАС-М [6], в которой традиции [I] подвергались глубокой ревизии и обновлению, предлагаемая система построена в рамках максимальной преемственности. Транслятор с языка ЛЯПАС взят из [5], почти целиком вошла в ПС базовая система [3], оттуда же вошли языки квазипараллельного [7] программирования и организации моделирования [8], некоторые программы анализа и синтеза однородных структур. Основной задачей, решаемой при разработке ПС и ее инструментального языка Р-ЛЯПАС, была автоматизация системного программирования. ПС-Р-ЛЯПАС допускает автоматическое построение систем общим объемом порядка I20000 команд, содержащих до 32-х сегментов.

2. Базовый уровень

Базовый уровень языка не очень отличается от того, что предложено в [I,2], тем не менее ниже дается его полное описание с новой, по нашему мнению, более подходящей точки зрения.

2.I. При описании синтаксиса будем пользоваться языком теории множеств. Установим следующее старшинство операндов: $\{x\}$, $\{\cap\}$, $\{\cup,\setminus\}$. В случае его нарушения употребляются фигурные скобки. Декартово произведение множеств будет записываться без специального символа (x), так что если A и B - множества и $a \in A$, то AB - их декартово произведение $A \times B$, а $aB \subseteq AB$, где aB - все слова длины 2, начинающиеся с a. Такая запись восприни-

мается как конкатенация, что вполне соответствует ее смыслу.

Пусть $\bar{A}$ - множество всех непустых слов алфавита A, $\bar{A}^n$ - множество всех слов длины не более n.

Если некоторый символ алфавита записывается несколькими буквами, например, "ввод", то при написании это обозначение подчеркивается "ввод", а при печати набирается жирным шрифтом.

Алфавит языка Р-ЛЯПАС показан в табл.I. Множество символов разбито на подмножества, обозначения которых даны в строке "тип", а смысл ясен из дальнейшего. Числа в табл.I и во всем последующем тексте восьмеричные. Каждому символу алфавита сопоставлен код, т.е. машинно-независимое целое число. В Р-ЛЯПАСе это число имеет 3 восьмеричных разряда и получается как сумма чисел, стоящих в графе к о д в строке и столбце табл.I, на пересечении которых находится символ. Код может быть сконструирован из логических и натуральных констант (Н), так что любая запись на языке Р-ЛЯПАС является объектом машинно-независимого программирования на Р-ЛЯПАСе. Следствием этого факта является машинная независимость подавляющей части программ ПС-Р-ЛЯПАС. При написании программы на Р-ЛЯПАСе коды могут употребляться вместо соответствующих символов, а также в некоторых конструкциях макроуровня языка. В этих случаях код подчеркивается, а при печати набирается жирным шрифтом.

2.2. О б ъ е к т программирования есть знаковая система, заданная набором своих атрибутов: имя, тип, адрес, структура, формат, значение, статус.

О п е р а н д есть объект или любой атрибут объекта, подвергающийся преобразованию или участвующий в нем как аргумент.

Объектами базового уровня Р-ЛЯПАСа являются я ч е й к и, к о м п л е к с ы, б у л е в ы в е к т о р ы и д е й с т в и т е л ь н ы е ч и с л а. Я ч е й к и суть ячейки памяти ЭВМ, к о м п л е к с ы - линейно упорядоченные множества ячеек, расположенные в памяти ЭВМ сплошным массивом и в соответствии с заданным линейным порядком. В частности, вся оперативная память ЭВМ есть специальный о п е р а т и в н ы й комплекс К. Остальная память ЭВМ представляет собой в н е ш н и й комплекс. Возможность работы с этими объектами отражает машинную ориентацию базового уровня.

Б у л е в ы в е к т о р ы и д е й с т в и т е л ь н ы е ч и с л а являются машинно-независимыми объектами, порожденными в ячейки и комплексы.

Таблица I

Внутренняя символика языка Р-ЯПАС

Тип	Оператор		$К_1$	$К_2$	П				Н				Ч
Код	000	040	I00	I40	200	240	300	340	400	440	500	540	700
0	,	;	А	А	а'	а	а	а'	0	40	I00	I40	а
I	§	апп	В	В	b'	b	b	b'	I				б
2	(	Ж	С	С	с'	с	с	с'	2				ц
3	)	Ж	D	D	d'	d	d	d'	3				д
4	!	→	Е	Е	е'	е	е	е'	4				е
5	↓	↔	F	F	f'	f	f	f'	5				ф
6	.	↕	G	G	g'	g	g	g'	6				г
7	Σ	↑	Н	Н	h'	h	h	h'	7				х
I0	Δ	▽	I	I	i'	i	i	i'	I0	50	II0	I50	и
II	Δ	$пч_{10}$	J	J	j'	j	j	j'	II				й
I2	о	←	К	К	k'	k	k	k'	I2				к
I3	о	⊢	L	L	l'	l	l	l'	I3				л
I4	→	⌐	М	М	m'	m	m	m'	I4				м
I5	←	ввод	N	N	n'	n	n	n'	I5				н
I6	зп	пчл	Р	Р	p'	p	p	p'	I6				п
I7	чт	пфл	Q	Q	q'	q	q	q'	I7				я
20	[	↑	R	α	r'	r	r	r'	20				р
2I	]	<	S	β	s'	s	s	s'	2I				с
22	⊥	>	Т	γ	t'	t	t	t'	22				т
23	П	сб	U	δ	u'	u	u	u'	23				у
24		рвб	V	ε	v'	v	v	v'	24				ж
25		:	W	ζ	w'	w	w	w'	25				в
26	буч	×	Х	η	x'	x	x	x'	26				ь
27	чкб	÷	Y	θ	y'	y	y	y'	27				ы
30	+'	+	Z	ι	z'	z	z	z'	30	70	I30	I70	з
3I	-'	-	К	λ	к'	к	к	к'	3I				ш
32	:'	Δ	Л	μ	л'	л	л	л'	32				э
33	×'	∨	Ф	ν	ф'	ф	ф	ф'	33				щ
34	#	⊕	Ш	ξ	ш'	ш	ш	ш'	34				ч
35	//	кок	Э	π	э'	э	э	э'	35				ъ
36	⊙	пчм	Ю	ρ	ю'	ю	ю	ю'	36				ю
37	⊗	пфм	Я	σ	я'	я	я	я'	37	77	I37	I77	я
Тип			$К_1$	Г	$П_1$		$П_2$		$Н_1$	$Н_2$			Ч

Рассмотрим подробнее атрибуты перечисленных объектов.

2.2.1. И м я есть знак, употребляемый для идентификации объекта, значение которого необходимо использовать или изменить (называя имя, имеют в виду значение). Множество базовых имен в Р-ЛЯПАСе конечно и представляет собой $К_1 \cup К_2 \cup П \cup Н \cup Ч$. В табл. 2 приведены различные множества имен, допустимых в программах на Р-ЛЯПАСе и построенных из базовых. В частности, можно построить

Т а б л и ц а 2

Множество базовых имен языка Р-ЛЯПАС

Название		Выражение
Константы	Натуральные	$Н = 0 \cup Н_1 \cup Н_2$ $Н_3 = Н_1 \cup Н_2$
	Восьмеричные	$В =]\overline{Н}[\ Н_1$
	Комплексы	$К_3 = \{C, D, E, F, L, P\}$
	Индексы	$И = П \cup Н$
Переменные	Числа	Ч
	Булевы векторы	$П = П_1 \cup П_2$
	Комплексы	$К = К_1 \cup (К_2 \setminus К_3)$
	Полные	$Л_2 = П \cup КИ$
	С полем	$Л_3 = Л_2\ [И] \cup Л_2[ИН]$
	Общего вида	$Л_1 = Л_2 \cup Л_3$
	Логические	$Л = Л_1 \cup Н \cup В \cup К_3И$
	Действительные	$Д = Ч \cup К_1И$
	Вводимые	$Э = Ч \cup П \cup К \cup АН \cup ВН$ $Э_2 = Э \setminus Ч$
	Выводимые	$Х = К_1 \cup К_2$
	Принимающие размерность	$Р = К; \cup П$
	Дающие размерность	$Ф = К; \cup Л$
	Именующие системные полюса	$П_3 = П_1 \setminus \{Ю', Я\}$

имена отдельных элементов комплексов (КИ), ячеек как элементов оперативного комплекса, частей булевых векторов, начинающихся с разряда И длиной Н разрядов ($Л_2$ [ИН]), отдельных разрядов ($Л_2$[И]) и констант (В). Имя любого объекта является одновременно именем ячейки (или их множества), в которую погружен этот объект. С каким конкретно объектом (ячейкой или погруженным в нее вектором) работает программа, в большинстве случаев не определяется и зависит от интерпретации. Более строгие правила на этот счет даны при описании операций.

2.2.2. Т и п объекта задается множеством операций (см. п.2.3) языка, в которых, согласно синтаксису, может быть упомянуто имя этого объекта. На базовом уровне языка выделяются три типа объектов: я ч е й к а (Я), л о г и ч е с к и й (Л) и д е й с т в и т е л ь н ы й (Д). Тип комплекса определен и отличен от типа Я, если в каждую его ячейку погружен объект одного и того же типа: Л или Д. В этом случае говорят о логических и действительных комплексах.

Заметим, что тип Я содержит в себе типы Л и Д, в том смысле, что любая операция над объектом Я допустима и над объектом типа Л или Д. Кроме того, из табл.2 видно, что $Л \cap Д = К_1И$, тем не менее в тексте программы всегда можно содержательно определить, к какому типу относятся элементы из $К_1И$.

2.2.3. А д р е с – минимальный порядковый номер элемента в подмножестве элементов оперативного или внешнего комплексов, в которое (подмножество) погружен соответствующий объект (или подобъект). Р-ЛЯПАС наследует адресацию объектов ЛЯПАСа, в которой адреса логических переменных (множество П) совпадают с их кодами. Адреса комплексов доступны и хранятся в специальном комплексе А: A_0 – адрес комплекса А, A_1 – адрес В и т.д. Таким образом, все комплексы являются п л а в а ю щ и м и. Кроме того, так как комплекс есть сплошной массив ячеек, то адрес i-го элемента j-го комплекса всегда равен $A_j + i$.

Адресация объектов, расположенных во внешней памяти, зависит от типа ЭВМ и числа внешних устройств. Общий принцип упорядочивания внешних устройств следующий. Внешний комплекс – продолжение оперативного, порядковые номера его элементов следуют за максимальным номером в К и чем они больше, тем больше номер и меньше быстродействие соответствующего внешнего накопителя.

2.2.4. С т р у к т у р а - это представление объекта в виде множества элементов и отношений между ними. Элементы, в свою очередь, могут быть объектами, способными дробиться, и т.д. до тех пор, пока в языке имеется такая возможность. Атомами в Р-ЛЯПАСе являются отдельные разряды и действительные числа (объекты типа Д).

Ячейка рассматривается как упорядоченное множество разрядов, мощность которого задается ЭВМ, а доступ возможен к любому разряду с помощью операции неограниченного сдвига (⇄ Л).

Структура элементарного логического объекта задается числом ρ его булевых компонент (размерностью). Компоненты логического вектора перенумерованы слева направо, начиная с нуля, $\rho \leq 40_8$. Размерность задается статическими операциями языка (см.табл.4), в противном случае она стандартна ($\rho = 40_8$), и сами такие объекты называются стандартными. В частности, все специальные комплексы ($К_2$) и натуральные константы (Н) стандартны.

Как следует из п.2.2.1, в Р-ЛЯПАСе доступна любая часть логического вектора, а наличие операций конкатенации, сборки и разборки (табл.4) обеспечивает доступ к любому подмножеству компонент логического вектора, причем нумерация компонент в любой части логического объекта начинается с нуля.

Структура комплексов доопределяется заданием мощностей, хранящихся в специальном комплексе В аналогично адресам. Все,что сказано выше о размерностях и частях операндов, относится и к элементам логических комплексов.

Итак, элементы большинства объектов Р-ЛЯПАСа являются операндами языка. Исключение составляют только действительные числа,которые неделимы.

2.2.5. Ф о р м а т есть описание способа погружения одного объекта в другой. В данном случае речь идет о погружении логических и действительных объектов в ячейки оперативного комплекса К, моделирующих ячейки памяти и регистры ЭВМ. Приняты следующие правила:

1. Числа имеют машинный формат.

2. Стандартные логические объекты размещаются в ячейках одинаковым образом. В БЭСМ-6 - в правом конце ячейки.

3. Прочие логические объекты размещаются в правом конце стандартных.

2.2.6. З н а ч е н и е - элемент предметной области знака (имени). Обычно в языках высокого уровня предметные области являются множествами математических объектов (целые, действительные, булевы, символы и т.д.), имена которых (множеств) одновременно являются именами типов объектов. В Р-ЛЯПАСе такое определение годится только для предметной области объектов типа Д, которая является множеством действительных чисел, представимых стандартным образом в данной ЭВМ.

Предметными областями объектов Я и Л являются все возможные наборы единиц и нулей, которые могут быть расположены в той области памяти, куда погружен соответствующий объект. Интерпретация значения оставляется за программистом. В частности, значение логической переменной размерности ρ может интерпретироваться как ρ-разрядный булев вектор или минимальный положительный вычет класса 2^{ρ}.

Такое определение предметной области приводит к высокой гибкости в использовании возможностей ЭВМ. Значения объектов могут меняться не только в результате прямых операций над ними, но и с изменением атрибутов (адреса, структуры, типа) или элементов структуры объекта. Доступны также элементы структуры сложного атома типа Д, так как Д$\subset$Я. О возможных злоупотреблениях такой гибкостью сказано в п.2.3.1.

Объекты, значения которых не могут изменяться, называются к о н с т а н т а м и. Значения натуральных констант (Н) совпадают с их именами, восьмеричных (В) - содержатся между скобками], [, причем левые нули могут быть опущены. Специальные комплексы из K_3 имеют следующие значения.

Элемент C_i содержит единицу в i-м разряде.

Комплексы D, E, F представлены в табл.3.

Элемент L_i специального комплекса L содержит i единиц в левом конце, если $i \le 40$, и $i-40$ единиц в правом конце, если $40 < i < 100$.

Специальный комплекс Р отводится для нужд программирующей системы и формируется при трансляции.

2.2.7. С т а т у с - правило доступа к объекту программирования. На базовом уровне Р-ЛЯПАСа все объекты доступны всем программам и имеют в них одни и те же имена. Иными словами, все объекты - с и с т е м н ы е. На макроуровне будет введена более гибкая система правил доступа.

Таблица 3

Специальные комплексы D, E, F

D_0	IOIO	IOIO	IOIO	IOIO	IOIO	IOIO	IOIO	IOIO
D_1	IIOO	IIOO	IIOO	IIOO	IIOO	IIOO	IIOO	IIOO
D_2	IIII	0000	IIII	0000	IIII	0000	IIII	0000
D_3	IIII	IIII	0000	0000	IIII	IIII	0000	0000
D_4	IIII	IIII	IIII	IIII	0000	0000	0000	0000
E_0	OIOI	OIOI	OIOI	OIOI	OIOI	OIOI	OIOI	OIOI
E_1	OOII	OOII	OOII	OOII	OOII	OOII	OOII	OOII
E_2	0000	IIII	0000	IIII	0000	IIII	0000	IIII
E_3	0000	0000	IIII	IIII	0000	0000	IIII	IIII
E_4	0000	0000	0000	0000	IIII	IIII	IIII	IIII
F_0	0000	0000	0000	0000	0000	000I	I000	0000
F_1	0000	0000	0000	0000	0000	000I	II00	0000
F_2	0000	0000	0000	0000	0000	000I	III0	0000
F_3	0000	0000	0000	0000	0000	000I	IIII	0000
F_4	0000	0000	0000	0000	0000	000I	IIII	I000
F_5	0000	0000	0000	0000	0000	000I	IIII	II00
F_6	0000	0000	0000	0000	0000	000I	IIII	III0
F_7	0000	0000	0000	0000	0000	000I	IIII	IIII
F_{10}	IIII	IIII	IIII	IIII	IIII	IIII	IIII	IIII
F_{11}	Единицы во всех разрядах ячейки ЭВМ							

2.3. О п е р а ц и е й называется любое преобразование операндов. В языке программирования определяется базовый набор операций, посредством которых выражаются все остальные преобразования. Поскольку для удобства этот набор обычно делается избыточным, то базовые операции также могут выражаться друг через друга. Учитывая это, назовем б а з о в о й о п е р а ц и е й языка минимальное слово этого языка, которому еще может быть поставлена в соответствие машинная программа. Очевидно, что набор базовых операций может обогащаться с появлением новых возможностей в языке ЭВМ или даже с разработкой новой версии транслятора.

2.3.I. Программа на любом языке содержит статическую и динамическую части. С т а т и ч е с к а я часть содержит предписания для транслятора и реализуется транслятором в порядке чтения программы. Обычно это описания объектов, действующие во всей области программы вплоть до считывания нового описания объекта того

же имени. В частности, таким объектом является сама программа. Динамическая часть содержит операции, выполняемые программой в порядке реализации.

2.3.2. Базовый уровень языка Р-ЛИПАС содержит два типа статических операций: полюса и согласование размерностей. Их описание дано в табл.4. Множество Р, упомянутое в табл.4, имеет вид $P = \{+\Phi, -\Phi, \Phi\}$, причем в вычислениях участвуют размерности операндов из $Д_1 \cup В$ и значения констант из Н.

Переменные из $П_3$, упомянутые в лок $П_3$, играют роль имен полюсов. Адреса этих меток хранятся в соответствующих элементах специального комплекса Q. Таким образом, эти переменные могут использоваться и в динамической части программы.

Статические операции нельзя использовать для динамического изменения значений (операндов и меток программы). Такими злоупотреблениями могут быть в частности:

а) изменение значений элементов комплекса Q или переменных $п'$ и $я'$, в результате которых последующие переходы в системные полюса будут неверными из-за нарушения базирования програмы;

б) изменение значений с помощью изменения размерностей операнда.

2.3.3. Особенностью динамических операций языка Р-ЛИПАС является тот факт, что большинство их преобразует некоторый "невидимый" текущий результат τ. Собственная переменная τ не отображается в языке и изменяет свое значение, размерность и тип под воздействием базовых операций в порядке их чтения слева направо или в порядке реализации, если встретились ветвления. Операции, в свою очередь, могут обрабатывать только τ определенного типа, и, следовательно, не всякая их последовательность есть программа.

В связи с этим синтаксис базового уровня состоит из нескольких разделов: уже описанного синтаксиса операндов, синтаксиса операций и синтаксиса графов переходов програми. Описание операций (табл.4), в свою очередь, содержит: наименование, соответствующее синтаксическое выражение, формулу для вычисления размерности выходного τ вида $f(\rho, \rho(Д))$ и пары Т из множества $\{Я, Д, Д_-\}$, где Я, Д, Д – указатели типа, если он определен, а знак "–" соответствует неопределенному τ. Левый элемент пары Т указывает, какой тип τ воспринимает данная операция, правый – выходной тип. Если Т отсутствует в описании операции, то это значит, что такая операция работает с τ любого типа и сохраняет его значение.

Таблица 4

Операции языка Р-ЛЯПАС

Статические операции языка Р-ЛЯПАС				
Название		Выражение	Размерность	Т
Полоса	собственный	§ $П_3$	40	Я,Я
	входной	§ 0	40	
	выходной	•	ρ	
	системный	лок $П_3$	40	
Согласование размерностей		⊗ $\overline{Р}$ → $\overline{Б}$ ⊗	-	Сохраняет тип

Динамические базовые операции языка Р-ЛЯПАС					
Наименование			Выражение	Размерность	Т
Засылка в т значений		логического вектора	Л	ρ (Л)	-,Я
		действительного числа	Д	ЭВМ	-,Д
		размерности	⊗ $\overline{Р}$ ⊗	40	-,Я
		случайного вектора	я	40	-,Л
		целой части частного от : Л	я	40	-,Л
Преобразования	типа	целая часть числа без знака	чиб	40	Д,Л
		переход к типу действительный	буч	ЭВМ	Л,Д
	значения текущего результата арифметические по модулю 2^{40} и логические	подсчет единиц (взвешивание)	▽	1 + [$\log_2 \rho$]	Л,Л
		номер левой единицы	⊢		
		нормализация	⊢	ρ	
		инверсия	¬		
		сдвиг влево	< Л		
		сдвиг вправо	> Л		
		конъюнкция	∧Л	min{ρ,ρ (Л)}	Я,Я
		дизъюнкция	∨Л		
		дизъюнкция с исключением	⊕Л		
		вычет по модулю Л	:Л		Л,Л

Таблица 4 (продолжение)

Наименование				Выражение	Размерность	Т
Преобразования значения текущего результата	арифметические по модулю 2^{40} и логические		сложение по модулю 2^{40}	$+Д$	$1 + \max\{\rho,\rho(Д)\} \le 40$	Д,Д
			вычетание по модулю 2^{40}	$-Д$	иначе 40	
			конкатенация справа	$=\!\!\rightarrow Д$	$\rho + \rho(Д) \le 40$	
			конкатенация слева	$\leftarrow\!\!= Д$	иначе 40	
			умножение по модулю 2^{40}	$\times Д$		
			старшие разряды произведения	$\dot{\times} Д$	$\rho + \rho(Д) - 40 > 0$; I	
			сборка разрядов, выделенных в Д	$\underline{сб}Д$	$\rho(Д)$	
			разборка (обратная сборка)	$\underline{разб}Д$		
		численные	сложение	$+'Д$	ЭВМ	Д,Д
			вычетание	$-'Д$		
			умножение	$\times'Д$		
			деление	$:'Д$		
	сдвиг ячейки ($\times 2^{Д-100}$)			$\rightleftarrows Д$	ЭВМ	Я,Я
Засылка значения τ	в действительный операнд			$\Rightarrow Д$	ЭВМ	Д,Д
	во множество логических операндов			$\Rightarrow(\bar{Д}_1)$		Д,-
Преобразование значений операндов	$\tau = Д_1$	в логический операнд		$\Rightarrow Д_1$	$\rho(Д_1)$	-,Я
		обмен значениями		$Д_1' \Leftrightarrow Д_1$		
		засылка пустого вектора		$o\, Д_1$		
		засылка вектора без нулей		$\bar{o}\, Д_1$		-,Д
		положительное приращение (+I)		$\Delta Д_1$		
		отрицательное приращение (-I)		$\bar{\Delta} Д_1$		
	$\tau = П$	перебор единиц	слева-направо	$Д_1 Ж Н_3 П$	$I + \log_2 \rho(Д_1)$	
		с выходом по нулю	случайный	$Д_1 Ж Н_3 П$		

Таблица 4 (продолжение)

Наименование			Выражение	Размерность	Т
Преобразование значений операндов		приписывание в конец комплекса	$(\bar{Э}_2) \to К$	ЭВМ	-,-
		извлечение конца комплекса	$К \to (\bar{Э}_2)$		
Запись во внешний накопитель			зп ППП		
Чтение с внешнего накопителя			чт ППП		
Ввод		через буфер ввода с рассылкой	ввод $\bar{Э}$;		
		непосредственно адресу ввода	ввод;		
Печать	τ	восьмеричная	пчл		
		десятичная	$пч_{10}$		
	комплекса	восьмеричная	К пчл		
		десятичная	К $пч_{10}$		
		символьная	К апш		
Переходы и полюсы программы		безусловный	$\to H_3$	Сохраняет размерность, значение и тип	
		по нулю: $\tau = 0$, $\tau < 0$, $\tau \geq 2^{40}$	$\circ\!\!\to H_3$		
		по единице: $\tau \neq 0$, $0 \leq \tau < 2^{40}$	$\mapsto H_3$		
		уход в подпрограмму	$\Rightarrow H_3$		
		возврат из подпрограммы	!		
		в системный полюс	$\downarrow П_1$		
Коммутаторы		по номеру левого нуля τ	$\circ\!\!\to [H_3]$	-	Л,-
		по номеру левой единицы τ	$\mapsto [H_3]$		
		по значению τ	$\to [H_3]$		
		начиная со значения τ	$\Rightarrow [H_3]$		

2.4. Программа на Р-ЛЯПАСе представляет собой согласованную последовательность операций, начинающуюся с входной метки § 0 и кончающуюся символом ".".

2.4.1. Статические операции вида § H_3 разбивают эту последовательность на предложения, и если каждой операции перехода $\{\to, \Rightarrow, \circ\!\!\to, \mapsto\}$ H_3, $Л_1Ж H_3П$, $Л_1Ж H_3П$ соответствует внутренний полюс § H_3, то программа согласована по по-

л ю с а м. Операции вида лок$П_3$ и $\downarrow П_1$ образуют системные входные и выходные полюса программы.

Полюса, перечисленные в табл.4, назовем с т а т и ч е с к и м и, а точки возврата (операции, следующие непосредственно после $\mapsto H_3$ и $\downarrow П_1$) - д и н а м и ч е с к и м и. Адреса динамических полюсов образуются в процессе реализации программы и запоминаются в магазине для $\mapsto H_3$ и в переменной p' для $\downarrow П_1$. Кроме того, полюса называются х о р о ш и м и, если первая исполняемая в этой точке операция допускает неопределенное τ (-), и п л о х и м и - в противном случае.

2.4.2. Построим граф-схему программы следующим образом.

I) Каждой операции ставится в соответствие вершина граф-схемы, помеченная соответствующей парой Т.

2) Возможный порядок следования операций в реализации программы отображается системой дуг, причем дуги, соответствующие операциям $\mapsto H_3$, и вершины, соответствующие !, помечаются. Большинство дуг этого графа будет отображать линейный порядок записи операций за исключением тех случаев, когда он обязательно нарушается переходами $\rightarrow$, $\mapsto$, $\downarrow П_1$, ! .

Коммутаторы $\{\rightarrow, \mapsto, \multimap, \mapsto\}[H_3]$ порождают дуги, идущие только к полюсам из списка H_3. Операции условного перехода порождают две дуги. Множество дуг, порождаемых операциями возврата (!), строится следующим образом. Каждая пара вершин, соответствующих $\mapsto H_3\alpha$, где α непосредственно следует за $\mapsto H_3$ в записи программы, соединяется дополнительной непомеченной дугой. Если в полученной дополненной граф-схеме есть непомеченный путь из $\S H_3$ в помеченную вершину (!), то последняя соединяется дугой с соответствующей вершиной α . После построения всех дуг возврата дополнительные дуги убираются.

3) Вершины полученной граф-схемы, соответствующие переходам и меткам, не имеют помечающей пары Т. Последняя строится следующим образом. Метки и возвраты (помеченные вершины) помечаются парой (Я,Я), а пара, соответствующая операции перехода, строится повторением второго элемента пары предшествующей вершины.

Граф-схема называется согласованной п о у п р а в л е н и ю, если в ней есть путь от любой вершины к основному (.) или системному ($\downarrow П_1$) выходам и если на любом пути от входа до выхода число помеченных дуг совпадает с числом помеченных вершин.

2.4.3. Отметки Т вершин граф-схемы можно рассматривать как отметки дуг. При этом начало дуги получает в качестве своей отметки правый элемент из Т той вершины, где эта дуга начинается, а конец дуги - левый элемент пары Т той вершины, где дуга кончается. Граф-схема будет согласованной п о т и п у τ, если все ее дуги имеют отметки, допустимые согласно табл.5. Так, в соответствии с табл.5, если дуга имеет входную отметку Я, то допустимы любые выходные, а если входная отметка "-", то допустима только такая же выходная отметка.

Т а б л и ц а 5

Допустимые отметки дуг граф-схемы программы

Вход	Выход			
	Л	Д	Я	-
Л	+		+	+
Д		+	+	+
Я	+	+	+	+
-				+

2.4.4. Используя графу "размерность" табл.4 и положение операций согласования размерностей, можно проследить также и за размерностью τ на любом пути граф-схемы. Граф-схема согласована п о р а з м е р н о с т и τ, если к моменту перехода в плохой полюс (перед операциями переходов в плохие полюса) собственная переменная имеет стандартную размерность.

2.4.5. Итак, последовательность операций согласована, если имеется согласованность по полюсам, по управлению, по типу τ и по размерности. Таким образом, понятие "программы" достаточно определено. Особый случай составляют коммутаторы, корректность которых зависит от значения τ. Последнее не должно порождать номера метки, выходящего за пределы списка $\bar{H}_3$.

Сложность проверки согласованности можно резко сократить, если проверять согласованность отдельных участков программы, начинающихся с меток и системных полюсов, а далее построить граф переходов и проверить его корректность, как это сделано в [2].

3. Макроуровень

Основными конструкциями макроуровня языка Р-ЛЯПАС являются м а к р о о п е р а ц и и и соответствующие им программные м о д у л и. Макрооперацией называется такое преобразование объектов, которое может быть описано некоторой программой базового уровня. Соответствующая программа, записанная для произвольных объектов заданного типа (вместо имен объектов употребляются фиктивные имена из множества Г табл.I), называется модулем.

Основные идеи иерархического программирования даны в [I-4]. Попытка формального описания синтаксиса макроуровня Р-ЛЯПАСа, сделанная в [2], привела к громоздким и туманным построениям, ничего не дающим для анализа правильности программ. Ниже мы попытаемся ввести основные понятия этого уровня достаточно строго и в то же время содержательно.

3.I. Понятие модуля имеет смысл, если программист "умеет видеть" модули (соответствующие макрооперации) еще до написания программы. Для анализа структуры модуля предположим, что мы "увидели" его уже в готовой внешней программе как некоторую сплошную последовательность операций, повторяющуюся для разных объектов, может быть, с некоторыми отличиями. Преобразуем эти участки в т е л о модуля, проведя в каждом из них следующие действия.

3.I.I. В начале поставим §0, а в конце "." .

3.I.2. Введем идентичные системы меток вида § H_3, начиная с § I, 2 и т.д., соответственно скорректируем все операции перехода. Если внутри тела модуля есть переходы во внешнюю программу, то соответствующие номера предложений заменяются на символы из Г (α, β и т.д.), а операции вида $\{\rightarrow, \mapsto, \multimap, \rightarrowtail\}\Gamma$ называются д о п о л н и т е л ь н ы м и в ы х о д н ы м и п о л ю с а м и модуля. Аналогично, если во внешней программе найдутся переходы в собственные полюса тела модуля, то соответствующие метки суть д о п о л н и т е л ь н ы е в х о д н ы е п о л ю с а модуля, а переходы к этим меткам во внешней программе – ее д о п о л н и т е л ь н ы е в н у т р е н н и е п о л ю с а.

3.I.3. Объекты, упомянутые не только внутри тела модуля (тела), но и во внешней программе, т.е. те объекты, над которыми действует соответствующая макрооперация, переименуем. Имена и в этом случае берутся из множества Г (см.табл.I) подряд. Заметим, что речь здесь идет не о случайном совпадении имен разных объектов в теле модуля и вне его, а об одних и тех же объектах, как бы они ни были названы. Выделенные таким образом операнды называются в н е ш н и м и, а оставшиеся непереименованными – в н у т р е н н и м и. Внешние операнды служат для настройки модуля на работу с данными объектами.

Иногда внешних операндов недостаточно для настройки модуля. Так бывает, если макрооперация допускает различные варианты, например, разные условия для некоторых своих действий. В таких случаях допустим, что имена из Г могут обозначать в ы р а ж е н и я,

составленные из символов входного алфавита. Операнды, входящие в эти выражения, также считаются внешними. Очевидно, что внутренние части выражений должны быть последовательностями операций, но на правом и левом концах могут находиться "обломки" операций и даже имен, продолжение которых находится в теле модуля. Таким образом, выражение не принадлежит базовому уровню языка, например, внешний предикат может иметь вид (Л∧Л↛) или (Л-Л↦).

3.I.4. Присвоим внутренним операндам, хранящим во всех экземплярах модуля промежуточные результаты, одни и те же имена.

3.I.5. Наконец, нам необходимо согласовать адреса и структуры внешних, внутренних входных и выходных объектов модуля.Если некоторое имя из Г заменяет имя комплекса, то допустимы выражения АГ, ВГ, СГ, обозначающие соответственно начало Г, мощность Г и тот элемент специального комплекса С, который соответствует комплексу Г, так что программа вида СГ⊢ ∨ I00 дает код этого комплекса.Согласование размерностей и использование их как аргументов достигается тем, что среди операндов, входящих в Р и Ю (см.табл.4),также можно вписывать фиктивные имена внешних объектов.

3.2. М а к р о с - слово, подставляемое в программу или модуль в том месте, где должно находиться тело соответствующего модуля, настроенное на работу в этой программе (модуле).

Это слово имеет следующий вид:

$$\#\, N_1 N_1\, п е р е ч е н ь\, //\,,$$

где N_1 - множество кодов (см.п.2.I), за исключением 034, соответствующего символу # ; п е р е ч е н ь - последовательность выражений, которые следует подставить в тело модуля на места, обозначенные символами из Г. Перечень упорядочен в порядке возрастания кодов символов из Г: α, β, γ ... и т.д. Выражение, подставляемое в перечень на место каждого символа Г, заключается в круглые скобки (,). Скобки можно опускать, если выражение содержит всего один символ, например, вместо (А) можно писать А. Если выражение представляет собой метку, то оно записывается в полной форме (§ H_3).

Таким образом, символ # - признак начала описания макроса , N_1N_1 - имя макроса, перечень задает настройку макроса, а символ // - признак конца описания макроса. Условимся скобки # и // считать о с н о в н ы м и в н у т р е н н и м и полюсами внешнего модуля.

Переходы в дополнительные входные полюса модуля имеют вид $\{\rightarrow, \mapsto, \multimap, \mapsto\}\,\#N_1N_1H_3H_3'$, где $\#N_1N_1$ – идентификатор макроса, H_3 – номер его употребления в программе (начиная с I), H_3' – номер его дополнительного входного полюса.

Следует учесть, что перечень принадлежит объемлющему модулю, а не макросу и, значит, в нем могут появиться внешние операнды этого модуля. Совершенно аналогично на месте меток в перечне могут появиться дополнительные входные полюса других макросов объемлющего модуля, т.е. выражения вида ($\#N_1N_1\,H_3H_3'$), или его выходные полюса вида (§ Г). В выражениях перечня макроса запрещается употребление макросов и операций § О и ".", а в теле модуля не могут употребляться макросы, соответствующие какому-либо объемлющему модулю (т.е. рекурсивные определения запрещены). При таком определении макросы и модули образуют и е р а р х и ю. Программа, не содержащаяся ни в каком модуле, называется г о л о в н о й и расположена н а г л у б и н е I, а модуль макроса, употребляемого на глубине i, располагается на глубине $i+I$.

3.3. Рассмотрим подробнее структуру модуля.

3.3.I. Прежде всего модуль представляет собой многополюсник. Сводка всех его полюсов приведена в табл.6. В ней символ $\rightarrow$ можно везде заменить элементами множества $\{\rightarrow, \mapsto, \multimap, \not\rightarrow, \not\mapsto\}$. Статические и динамические полюса отмечены в специальной графе буквами С и Д соответственно. Внутренние полюса замыкаются на соответствующие основные и дополнительные входы внутренних модулей, а дополнительные – на собственные полюса внешнего модуля.

Заметим, что для использования макроса программисту необходимо знать лишь о наличии дополнительных внешних полюсов, поскольку они влияют на организацию внешнего модуля (входы) и настройку в нем макроса (перечень). Системные же полюса скрыты от пользователя, и, вообще говоря, их использование выходит за рамки иерархического программирования (см.п.4.2).

3.3.2. При построении п.3.I единого экземпляра модуля внутренние операнды получили имена вне всякой связи с именами операндов внешнего модуля. Для того чтобы программирующая система была в состоянии "развязать" модули по именам внутренних операндов с минимальными затратами памяти, ей необходимо сообщить информацию о статусе соответствующих объектов. На макроуровне языка вводятся четыре статуса объектов: системный, собственный, рабочий и служебный.

Таблица 6

Система полюсов модуля

Классификация			Основные	Дополнительные
Собственные		С	§ H_3	
		Д	→ H_2 полюс	
Внутренние		С	#, //	→#$N_1N_1H_3H_3$
		Д		↦#$N_1N_1H_2H_3$ полюс
Внешние	вход	С	§ О	§ H_3
	выход	С	•	→ Г
		Д		↦ Г полюс
Системные	вход	С	<u>док</u> $П_2$	
	выход	Д	↓$П_1$ \ <u>я</u> полюс	
		С	↓<u>я</u>	

С и с т е м н ы й объект является общим для некоторого множества модулей и имеет во всех этих модулях одно и то же имя. Он играет роль скрытого внешнего операнда и необходим при взаимодействиях модулей через системные полюса.

С о б с т в е н н ы й объект определяется внутри модуля, может быть преобразован этим модулем, но сохраняется за его пределами, и, следовательно, имя собственного объекта может быть упомянуто только в теле данного модуля. Наличие собственных объектов позволяет не выносить в перечень внешних операндов объекты, не существенные для понимания функций и использования макросов.

Р а б о ч и й объект существует только в теле данного модуля и не сохраняется за его пределами, где его имя (и место в памяти) может быть использовано. Однако если в теле модуля встречаются макросы, то рабочий объект не должен меняться при их работе, если он, конечно, не является выходным внешним операндом макроса. Имя рабочего объекта не может быть упомянуто в теле некоторого внутреннего модуля как его внутреннее имя.

С л у ж е б н ы й объект отличается от рабочего тем, что его существование не обязательно при реализации внутренних модулей и служебные имена могут как угодно пересекаться. В частности, если в теле модуля нет макросов, то нет и рабочих объектов.

Статус объекта можно определить более строго, рассмотрев его с л е д [9], т.е. множество динамических операций, при реализации

которых этот объект должен существовать. На граф-схеме программы эти операции образуют множество путей от операций присвоения значения какой-либо части объекта или его атрибутам (адрес, мощность комплекса) до тех операций, где какие-либо части или атрибуты объекта используются как аргументы. След, очевидно, распадается на три части:

F - множество операций, изменяющих объект как функцию;

A - множество операций, использующих объект как аргумент;

S - множество операций сохраняющих объект.

Системный объект имеет след, пересекающий внешние полюса нескольких модулей и такой, что его части F и A находятся в разных модулях. След собственного объекта отличается от системного тем, что его части F и A сосредоточены в теле данного модуля. След рабочего объекта пересекает только собственные и внутренние полюса модуля, а служебного - только собственные.

3.3.3. Полное описание модуля состоит из паспорта, содержащего описание модуля, и тела, оформленного по правилам п.3.I. Оно имеет следующий вид:

$$\# N_1N_1N\ N\bar{Я};\bar{Я};\ \bar{Э}_1;\ \bar{Э}_1$$
$$\S\ 0\ \text{ТЕЛО}.$$

Здесь $\# N_1N_1$ - идентификатор макроса, N - произвольный код, $\bar{Я};\bar{Я};\bar{Э}_1;\bar{Э}_1$ - списки системных, собственных, рабочих и служебных объектов соответственно:

$$Я = К_1 \cup К_1[\bar{Н}] \cup П \cup Ч \cup -, \qquad Э_1 = К_1 \cup П \cup Ч \cup - \quad .$$

Знак "-" означает пустой список и может быть пропущен.

Конструкция $К_1[\bar{Н}]$ употребляется в том случае, когда соответствующий системный или собственный комплекс имеет фиксированную максимальную мощность $\bar{Н}$ и она известна программисту при написании модуля. В этом случае программирующая система сама расположит этот комплекс в памяти ЭВМ.

Если в теле модуля встречаются имена, не специфицированные в списках статуса, то программирующая система не подвергает их переименованию и не защищает от пересечений. Такая ситуация может встречаться при использовании модулей системы ЯШАС.

При вводе в ЭВМ модуль представляет собой последовательность кодов, упакованных в ячейки так, что паспорт ($\# N_1N_1\ldots$) и тело модуля ($\S\ 0\ldots$) упаковываются с левого конца ячейки. Это отражено в записи соответствующих синтаксических конструкций с новой строки.

3.4. Любая программа, записанная на языке Р-ЛЯПАС, является модулем. Это относится и к головной программе, которая не может иметь внешних операндов и дополнительных полюсов и которой, тем не менее, можно поставить в соответствие макрос вида $\#И_1И_1//$.

4. Системный уровень

Макроуровень позволяет накапливать в архиве ПС небольшие модули и строить из них программы, общее число меток в которых не превышает 177_8, причем соответствующая машинная программа должна целиком помещаться в памяти ЭВМ. Для этой цели достаточно "наивной" компиляции, т.е. подстановки настроенного тела модуля на место соответствующего макроса. Макроуровень не освобождает программиста от выделения модулей, но обеспечивает выразительные языковые средства и автоматическую компиляцию.

Системный уровень предназначен для представления больших программ и систем программ, которые не умещаются в памяти ЭВМ и требуют сегментации. Как и макроуровень, системный уровень языка не освобождает программиста от решения задачи сегментации, но дает средства описания принятого решения и автоматизирует реализацию сегментированной программы или системы программ. Этими средствами являются системные полюса, системные объекты, сегменты и сопрограммы. Первые два понятия были введены на базовом и макроуровнях с целью обеспечения ручного системного программирования. Сегменты и сопрограммы полностью используют эти средства как в явном, так и в неявном виде. С другой стороны, в них отсутствуют дополнительные входные полюса.

4.1. С е г м е н т о м называется модуль, не имеющий дополнительных внешних полюсов и вынесенный за пределы внешней программы. Соответствующий макрос может входить в модуль, не являющийся сегментом. По способу загрузки модуля в память ЭВМ различаются мягкие и жесткие сегменты.

М я г к и й сегмент выносится за пределы оперативного комплекса во внешнюю память. В тот момент, когда программа обращается к мягкому сегменту, последний замещает ее в оперативном комплексе, а после исполнения вытесняется своей головной программой. Дальнейшее выполнение головной программы происходит с соответствующего внутреннего полюса // . В момент своего выполнения мягкий сегмент может быть в свою очередь вытеснен каким-либо внутренним сегментом и т.д. по известной схеме обхода дерева.

Единственная форма обращения к мягкому сегменту имеет вид

$$\#\#N_1N_1\ п\ е\ р\ е\ ч\ е\ н\ ь//\ ,$$

причем перечень не должен содержать меток.

Жесткий сегмент занимает дополнительное место в оперативной памяти и вытесняется только тогда, когда в нем самом встретился мягкий сегмент. Головная программа системы в последнем случае остается на своем месте. Употребляется жесткий сегмент в тех случаях, когда можно пожертвовать памятью в целях достижения большого быстродействия или если обращение к соответствующему макросу реализуется очень интенсивно. Обращение к жесткому сегменту отличается от предыдущего случая наличием дополнительного символа $\#$, а именно:

$$\#\#\#N_1N_1\ п\ е\ р\ е\ ч\ е\ н\ ь//\ .$$

Предлагаемый аппарат позволяет гибко управлять загрузкой крупных программ в память ЭВМ, но отсутствие дополнительных полюсов доводит жесткость иерархии модулей до предела.

4.2. Сопрограммами называются модули, взаимодействующие через системные полюса и системные операнды. Для этого сопрограммы должны находиться в оперативной памяти одновременно хотя бы в момент взаимодействия.

Это системное средство полезно при создании замкнутых систем макросов, образующих своего рода макроязыки. В таких случаях один модуль (сопрограмма), используемый как жесткий сегмент, выполняет действия по заявкам резидентных макросов, употребляемых в различных точках программы. Примером может служить язык моделирования систем параллельных процессов [7], содержащийся в архиве системы Р-ЛЯПАС.

Более тонкой является возможность использования сопрограммы без ее прямого упоминания, где бы то ни было в программе пользователя. Примером подобного рода являются процедуры супервизоров, предоставляемые пользователю со стороны любой операционной системы. В системе Р-ЛЯПАС такая сопрограмма автоматически появляется при использовании мягких сегментов. Вместо соответствующего макроса в программе записывается выход в системный полюс сопрограммы исполнитель, которая и организует смену сегментов и возврат в динамический полюс после исполнения макроса.

Если макрос не может быть выполнен без обращения к сопрограмме, то в паспорте соответствующего модуля указывается идентифика-

тор этой сопрограммы:

$$\# \; N_1 N_1 \text{NN } \bar{Я}; \; \bar{Я}; \; \bar{Э}_1; \; \bar{Э}_1$$
$$\# \# \# N_1 N_1$$
$$\S \; 0 \ldots$$

Это обеспечивает ее автоматическую загрузку в оперативный комплекс.

З а к л ю ч е н и е

Описанный выше язык предназначен в основном для внутреннего употребления в программирующей системе и обеспечивает преемственность программирования. Он является входным только для первой очереди ПС-Р-ЛЯПАС. Входной язык дальнейшего расширения системы должен, конечно, иметь алфавит внешних устройств ЭВМ. Было бы тривиально сопоставить каждому символу алфавита Р-ЛЯПАС некоторую конструкцию из знаков алфавита входных устройств ЭВМ. В идеале вторая очередь ПС должна иметь язык, обладающий следующими дополнительными свойствами:

1) возможностью работать с символами и строками символов, что соответствует работе с кодами;

2) наличием иерархических средств описания структур сложных объектов (макрообъектов) и архива макрообъектов;

3) мнематическим именованием, а не нумерацией макросов и макрообъектов;

4) наличием в языке метасимволов для описания синтаксиса макросов и макрообъектов.

Реализация программы второй очереди приведет к построению языка с открытым синтаксисом.

Л и т е р а т у р а

1. ЗАКРЕВСКИЙ А.Д. Описание языка ЛЯПАС.-В кн.: Логический язык для представления алгоритмов синтеза релейных устройств. М., Наука, 1966, с.7-38.

2. ВОРОБЬЕВ В.А. Р-ЛЯПАС - базовый язык моделирования цифровых систем.- В кн.: Вычислительные системы. Вып.39, Новосибирск, 1970, с.67-80.

3. ВОРОБЬЕВ В.А., ПЕТРОВА Э.А. Программирующая система ЛЯПАС-Р-ЛЯПАС для "Минск-22" (инструкция по эксплуатации). - В кн.: Вычислительные системы. Вып.59. Стандартные программы обработки информации. Новосибирск, 1974, с.162-179.

4. ЗАКРЕВСКИЙ А.Д. Алгоритмы синтеза дискретных автоматов. - М.: Наука,1974. - 511 с.

5. ЛЕВАШНИКОВ А.А. Система ЯПАС-70 для БЭСМ-6. Инструкции пользователю и программисту.-Томск,1975.(Сиб.физ.-техн.ин-т.)

6. ЗАКРЕВСКИЙ А.Д., ТОРОПОВ Н.Р. Программирующая система ЯПАС-М. - Минск: Наука и техника,1978.- 239 с.

7. ВОРОБЬЕВ В.А. Моделирование системы параллельных процессов на Р-ЯПАСе. - В кн.: Вычислительные системы.Вып.51. Новосибирск,1972, с.82-95.

8. ВОРОБЬЕВ В.А., КОРНЕЕВ В.В. Метод статистических испытаний при исследовании характеристик осуществимости. - В кн.: Вычислительные системы. Вып.60. Вопросы теории и построения вычислительных систем. Новосибирск, 1974, с.70-83.

9. БЫКОВА С.В., ИВОЛГА В.П. Выделение свободных операндов в программах на ЯПАСе. - В кн.: Материалы по системе автоматического программирования ЯПАС. Томск. Вып.6, с.3-14.

Поступило в ред.-изд.отд.
26 июня 1979 года