В.И.Гололобов Б.Г. Чеблаков Г.Д.Чинин

Описание языка ЯРМО

МАКРОСРЕДСТВА

248

ПРЕПРИНТ

НОВОСИБИРСК

Академия наук СССР Сибирское отделение
Вычислительный центр

В.И.Гололобов, Б.Г.Чеблаков, Г.Д.Чинин

ОПИСАНИЕ ЯЗЫКА ЯРМО

МАКРОСРЕДСТВА

Препринт

248

Новосибирск 1980

АННОТАЦИЯ

Описывается язык высокого уровня, машинно-ориентированный на ЭВМ БЭСМ-6 и предназначенный для написания комплексов системных программ. Во второй части описания определяются средства, основанные на открытых подстановках, и возможности прямого доступа к ЭВМ.

7. ОТКРЫТЫЕ ПОДСТАНОВКИ

Кроме структур данных, задаваемых описанием переменных, в языке имеется механизм, позволяющий конструировать произвольные структуры данных путем программирования алгоритмов доступа (см. 7.2). Процедурный аппарат языка (см. 5.2) может быть расширен путем задания различных классов процедур, которые в отличие от процедур, описанных в 5.2, называются квазипроцедурами (см. 7.3). Механизмы доступов и квазипроцедур основаны на идее открытых (текстовых) подстановок (см. 7.I).

7.I. Тексты

```
описание-текстов ::=
    общие ? тексты (описание-текста/;):
    виртуальные тексты (описание-виртуального-текста/;)
описание-текста ::= имя%текста%
    ( ( ( имя%формального параметра текста% /, ) ) )?
    ="текст"
описание-виртуального-текста ::=
    имя%текста% (* анкета%текста% )?
раздел-анкеты%текста% ::= числпар = константа
вызов-текста ::= имя%текста% ( ( (текст/,) ) )?
первичное (7.2) ::=
    вызов-текста : имя%формального параметра текста%
текст ::= предложение
```

Предполагается, что открытые подстановки совершаются перед исполнением программы.

Процесс преобразования исходной программы, который приводит к эквивалентной программе, не содержащей текс-

тов, можно описать следующим образом. В каждом месте вызова некоторого текста совершается соответствующая открытая подстановка. Этот процесс кончается в тот момент, когда полученная после очередной подстановки, программа уже не содержит ни одного вызова текста. После этого исключаются все описания текстов. Полученная программа готова к исполнению. Сама подстановка текста совершается в два этапа. Сначала все использующие вхождения формальных параметров в описании текста заменяются на соответствующие фактические параметры, причем эти подставленные фрагменты во всех дальнейших возможных преобразованиях и исполнениях рассматриваются в контексте вызова текста. Преобразованное подобным образом тело текста подставляется вместо вызова текста, опять-таки с учетом контекста, но теперь уже контекста описания текста.

Таким образом, можно выделить три особенности открытых подстановок языка ЯРМО.

I) Подстановки совершаются с учетом контекста, т.е. смысл, придаваемый объектам, встречающимся в описании текста, сохраняется при подстановках.

2) Тело текста имеет некоторую синтаксическую структуру. Язык предлагает в качестве такой структуры предложение . Предполагается, что при конкретных реализациях могут быть предложены дополнительные варианты.

3) Место вхождения вызова текста тоже синтаксически определено. Язык предлагает для этого позицию первичного. Конкретная реализация может расширить множество позиций вхождений.

Особенности, связанные с виртуальными текстами, аналогичны случаю с процедурами (см. 5.3).

7.2. Структуры данных

```
описание-доступов ::=
    общие ? доступы (описание-доступа/;):
    виртуальные доступы
    (описание-виртуального-доступа/;)
```



```
описание-доступа ::= имя%доступа%
    ([(имя%формального параметра доступа%/,)])?
    =("текст"→ "текст" :
    "текст"→ : "текст" :→ "текст")
описание-виртуального-доступа ::=
    имя%доступа% (* анкета%текста%)
переменная ::= имя%доступа% ( (текст /,) )?
первичное (7.3) ::= имя%формального параметра доступа%
```

Структуры, задаваемые описанием векторов (см. 2.3), предоставляют программисту некоторое стандартное средство для организации данных. При этом действия, связанные с выборами элементов векторов и их изменениями, выполняются неявно и должны быть обеспечены каждой конкретной системой программирования.

Кроме таких стандартных структур, язык представляет возможность программного построения произвольных структур данных с явным указанием действий по доступу к элементам структуры, возможными проверками и т.п.

Возможность программного задания структуры данных заключается в описании алгоритма доступа. При этом использующее вхождение имени доступа с соответствующим набором параметров возможно в позиции переменной. В описании алгоритма доступа в общем случае могут присутствовать два различных варианта алгоритма. Один вариант используется для выборки значения (чтения), второй - для изменения значения (запись). Возможны четыре случая при описании доступа.

I) "текст%чтения%"→ "текст%записи%"

Если переменная встречается в контексте первичного, то происходит подстановка текста чтения. Если же переменная встречается в позиции адреса засылки, то происходит подстановка текста записи, причем заменяемый фрагмент наряду с переменной включает в себя и стрелку.

2) "текст%чтения и записи%"→

Приведенная запись является сокращением записи для первого случая при условии, что текст записи отличается от текста чтения только предшествующей стрелкой засылки.

т.е. любое вхождение переменной, соответствующей данному описанию, заменяется на данный текст.

3) "текст%чтения%"

Этот случай разрешает только чтенке, т.е. вхождение переменной в позицию адреса засылки недопустимо.

4) ->"текст%записи%"

В этом варианте появление переменной допустимо только в позиции адреса засылки.

Сама подстановка совершается таким же образом, что и подстановка текстов (см. 6.I). Вопросы, связанные с виртуальными описаниями, трактуются так же, как и в случае с процедурами (см. 5.3).

I.3. Классы квазипроцедур

Процедурный аппарат, описанный ранее (см. 5.2), предполагает некоторую стандартную схему реализации, которая может оказаться в отдельных случаях неудовлетворительной.

Механизм классов квазипроцедур позволяет описывать более узкие классы процедур с программным заданием их реализации, учитывающим конкретную специфику. Если при описании процедурного блока (см. 5.2) предполагалось, что инициирование и завершение его работы обеспечивается стандартным образом, то в случае квазипроцедуры ее функционирование должно быть обеспечено соответствующим описанием класса. С помощью описания класса можно задавать произвольные спссобы передачи параметров. Ниже приводится синтаксис описаний и использований квазипроцедур.

```
описание-классов ::=
    общие ? классы (описание-класса/;):
    виртуальные классы (описание-виртуального-класса/;)
описание-класса ::= имя%класса%
    ( ( (имя%формального параметра класса%/,) ) )?
    ="текст%программирования вызова%"
    :"текст%программирования входа и выхода%"
```



```
описание-виртуального-класса ::=
    имя%класса% (* анкета%текста% )?
описание-квазипроцедур ::=
    общие ? квазипроцедуры (описание-квазипроцедуры/;):
    виртуальные квазипроцедуры
    (описание-виртуальной-квазипроцедуры/;)
описание-квазипроцедуры (8.I) ::=
    имя%квазипроцедуры% * (анкета%квазипроцедуры%)=
    формула
описание-виртуальной-квазипроцедуры (8.I) ::=
    имя%квазипроцедуры% * (анкета%квазипроцедуры%)
раздел-анкеты%квазипроцедуры% (8.I) ::= имя%класса%
    ( ( (текст%фактического параметра класса%/,) ) )?:
    ( чиспар : размпар ) = константа
первичное (8.6) ::=
    имя%формального параметра класса% :
    параметр ( константа%номер параметра% )? :
    повторитель: вызов-квазипроцедуры: тело
повторитель ::= <<текст>> ([константа : константа])?
вызов-квазипроцедуры ::=
    имя%квазипроцедуры% ( ( (текст
    %фактического параметра квазипроцедуры%/,) ) )?
```

Основное различие между блоками квазипроцедуры и процедуры заключается в следующем. При исполнении процедурного блока неявно предполагается, что перед исполнением собственно блока исполняются некоторые действия, определенные стандартным образом, которые вместе с неявными действиями, выполняемыми при завершении исполнения процедуры, обеспечивают функционирование процедурного аппарата таким образом, что становится возможным рекурсивное обращение, осуществление возврата в нужное место и т.д. Таким же образом исполняются скрытые действия при вызове процедуры, связанные с засылкой параметров и передачей управления на тело вызываемой процедуры. Программирование всех этих скрытых действий в каждой конкретной реализации языка выполняется транслятором. В случае же квазипроцедур все эти действия зада-

ются в программе явно посредством описания соответствующего класса, и, следовательно, согласование этих действий возлагается на автора программы. Указание размера поля параметров (размпар) вызывает отведение соответствующего участка памяти, необходимого для размещения фактических параметров квазипроцедуры.

Блок, получающийся из формулы в описании квазипроцедуры и называемый телом, преобразуется в итоговое предложение следующим образом.

I) Вхождения формальных параметров класса в правом тексте описания класса заменяются на фактические параметры из обращения к классу, заданному в анкете квазипроцедуры.

2) В преобразованный таким образом текст вместо символа тело подставляется тело квазипроцедуры. Все открытые подстановки совершаются в соответствии с правилами подстановки текстов (см. 7.I.).

Вызов квазипроцедуры перед его исполнением преобразуется следующим образом.

I) Все вхождения формальных параметров класса в левый текст из описания класса заменяются на соответствующие фактические параметры, взятые из анкеты соответствующей квазипроцедуры.

2) Все повторители переписываются путем дублирования. При каждом дубле к символам параметр , содержащимся в повторителе, приписывается номер в квадратных скобках, если последний не указан явно программистом. Номера пробегают значения из диапазона, указанного после повторителя заданием левой и правой границы, заключенных в квадратных скобках. Если эти границы отсутствуют, то количество дублей определяется количеством параметров вызываемой квазипроцедуры.

3) Вместо символов параметр с соответствующим номером, входящих в преобразованный на первом и втором этапах левый текст класса, подставляются фактические параметры из вызова квазипроцедуры. Полученный в результате описанных преобразований текст подставляется на место вызова квазипроцедуры.

7.4. Примеры

Ряд рассматриваемых ниже примеров не включает в себя примеров на доступы и классы квазипроцедур. Они будут приведены в следующей главе.

7.4.I. Влияние контекста на открытые подстановки

В приводимой ниже таблице содержимым левой колонки является текст блока, включающего в себя описание и вызов некоторого текста, а содержимым правой - текст этого же блока после осуществления соответствующей открытой подстановки. Здесь же стрелками указаны связи между использующими и определяющими вхождениями идентификаторов.

исходная программа	преобразованная программа
начало переменные X; У; Z ; текст T (*I*,*J*)="X+У+*J*+*I*"; вход I→X; 2→У; 3→Z; начало переменные Z ; X; У; вход 0→X→Z; I→У; T(X,Z); конец конец	начало переменные X; У; Z ; вход I→X; 2→У; 3→Z; начало переменные Z ; X;У; вход 0→X→Z; I→У; X+У+Z+X конец конец

В результате исполнения программы будет выработано значение 3.

7.4.2. Несколько примеров текстовых подстановок

Фрагмент программы	Результат исполнения

начало **переменные** X; У; Z ;
  текст T (*I*) ="*I*<У";
вход 0 -> X; 2 -> У;
  T(X) -> Z ;
  начало **переменные** X; У;
  вход 3 -> X; 5 -> У;
  T(X) = Z
  конец
конец
	ложь

начало **переменные** X; У;
  текст T (*I*) ="*I* ∨ У";
вход I -> X; 2 -> У;
  начало **переменные** X; У;
  вход **истина** -> X;
  ложь -> У;
  T(X);
  конец
конец
	не определено

начало **переменные** X; У;
  процедура Z = X x У;
  текст T (*I*) ="Z+ X +*I*" ;
вход 2 -> X; 3 -> У;
  начало **переменная** Z ;
  процедура X = У x Z
  вход 3 -> Z ; 2 -> У;
  T(X);
  конец
конец
	в I4

Для эффективного использования средств языка, описываемых в данной главе, может потребоваться знание ЭВМ БЭСМ-6, а также некоторых деталей конкретной реализации языка. Для непосредственного обращения к памяти путем прямой (см. 8.I) и косвенной (см. 8.2) адресации необходимо знать структуру оперативной памяти и ее разбиение, определяемое реализацией для продукта трансляции любой определенной программы. Для непосредственного использования команд (см. 8.4) требуется знание системы команд и основных характеристик БЭСМ-6. В этой же главе вводятся новые операции, обеспеченные системой команд БЭСМ-6. Для сегментирования больших программ вводится специальное средство (см. 8.5), которое уточняется реализацией языка.

Возможно программное обращение к адресам и другим атрибутам объектов (см. 8.6), набор которых может быть расширен при конкретной реализации.

Язык предлагает средство для указания транслятору выражений, которые следует вычислять при трансляции с целью получения более эффективного объектного кода (см. 8.7).

8.I. Использование особенностей адресации памяти

Тот факт, что адрес оперативной памяти БЭСМ-6 определяется смещением и индексом, отражен в языке конструкцией адресной пары, которая является одной из альтернатив адреса.

```
адресная-пара ::=
    ⟨ (константа%индекс%,)? константа%смещение% ⟩
адрес ::= адресная-пара
```

Адресная пара, заданная только смещением, определя-

ет постоянный адрес. Полная адресная пара во время исполнения программы может определять различные адреса в зависимости от содержимого индексного регистра, т.к. адрес определяется как сумма смещения и значения регистра. Если пара встречается в позиции адреса засылки, то исполнение засылки вызовет изменение содержимого ячейки с указанным адресом, причем новое значение явится результатом засылки. В позиции первичного адресная пара в качестве значения поставляет содержимое ячейки с соответствующим адресом.

Кроме такого явного использования в формулах адресная пара может быть связана с переменными при их описании.

```
описание-переменной ::=
    (адресная-пара=)?имя%переменной%
    ( [константа] )?( : запись)?(=(формула /.) )?
```

Явное указание пары при описании означает, что адрес начальной ячейки области памяти, отводимой под объект, указан парой. При этом следует учитывать, что во время исполнения фактический адрес, выраженный парой, может измениться при изменении значения индекса. Именно это свойство пары может потребоваться при желании указать с помощью пары адресацию для всех локальных объектов динамических блоков при описании квазипроцедур и процедур.

```
описание-процедуры ::=
    (адресная-пара=)?имя%процедуры%
    ( ( (формальный-параметр %процедуры% /,) ) )?=
    формула
описание-виртуальной-процедуры ::=
    (адресная-пара=)?имя%процедуры%
    ( * анкета%виртуальной-процедуры% )?
описание-квазипроцедуры ::=
    (адресная-пара=)? имя%квазипроцедуры%
    * анкета%квазипроцедуры% = формула
```



```
описание-виртуальной-квазипроцедуры ::=
    (адресная-пара=)? имя%квазипроцедуры%
    * анкета%квазипроцедуры%
```

Наличие пары, связанной с динамическим блоком, позволяет пересмотреть правило определения области доступности (в сторону расширения) для переменных динамических блоков. Ниже под динамической переменной понимается переменная, адрес которой зависит от индекса. Эта зависимость может быть указана явно с помощью пары, а у переменных процедуры, не связанной с парой, такая зависимость существует неявно и известна для каждой конкретной реализации.

Область доступности для динамической переменной в некотором блоке получается исключением из этого блока тех вложенных блоков, у которых индекс пары (не равный нулю) совпадает с индексом динамической переменной.

Наличие индексных регистров позволяет расширить набор структур управления за счет введения еще одного цикла и нового типа условия.

```
циклическое ::=
    по константа
    ( раз предложение)?
    цикл предложение
    ( иначе предложение)?
    все
условие ::= предложение
    ( ирноль константа : ирненоль константа )?
```

Семантика по-цикла описывается следующим образом. Сначала исполняется раз-предложение, результат которого указывает количество итераций цикл-предложения. Цикл организуется с помощью индексного регистра, указанного по-константой. Для этого он инициализируется результатом раз-предложения, представленным в дополнительном коде, причем при каждой новой итерации его содержимое уменьшается на единицу. Таким образом, при нормальном

завершении цикла содержимое индексного регистра становится равным нулю. Отсутствие раз-предложения приводит к отсутствию принудительной инициализации индексного регистра. Значением по-цикла при отсутствии иначе-предложения является результат цикл-предложения, полученный при последней итерации, в противном случае результат поставляется иначе-предложением.

Результатом исполнения условия, оканчивающегося на ирноль константа (ирнеполь константа),является значение истина в том случае, когда содержимое индексного регистра равно нулю (не равно нулю, соответственно). Номер индексного регистра, как и ранее, указывается константой. При этом результат исполнения если-предложения сохраняется.

8.2. Косвенность

```
косвенность ::= первичное + (адресная-пара:
    имя%константы или переменной% :
    имя%записи% (. селектор-записи)?)?
адрес ::= косвенность
```

Адрес, определяемый с помощью конструкции косвенности, получается как результат суммы значения первичного с адресом, указанным или адресной парой, или именем объекта, занимающего память. В последнем случае имеется в виду адрес первой ячейки объекта. Пару вида /0/ можно опускать.

Если косвенность встречается в позиции первичного, то значением этого первичного явится содержимое ячейки с полученным адресом. В том случае, когда косвенность встречается в позиции адреса засылки, содержимое по этому адресу меняется, и новое значение ячейки становится результатом исполнения засылки. Из синтаксиса косвенности и из того, что адрес является одной из альтернатив первичного, следует, что возможно несколько уровней косвенности. На адрес, полученный с помощью

косвенности, можно наложить запись. При этом начальная ячейка разбиения записи совмещается с полученным адресом. Содержимое поля, указанного селектором записи, является значением косвенности и может быть изменено, если косвенность встречается в позиции адреса засылки. Результатом исполнения засылки будет являться новое содержимое начальной ячейки разбиения данного поля.

8.3. Операции

Ниже приводится синтаксис и семантика операций, не вошедших в таблицу из пункта 3.I.

```
операция (8.4) ::= вчаб : вчабр : вчоб : вчобр : нтж :
    слц : чед : нед : сдв : сдвп : сдвл
```

Семантика операций дается ниже в виде таблицы (так же, как и в 3.I).

интерпретация значений операндов и результата	операция	результат
целое	вчаб	разность абсолютных величин операндов
	вчоб	обратная разность
рациональное	вчабр	разность абсолютных величин
	вчобр	обратная разность
произвольное	нтж	поразрядная сумма по модулю 2
	слц	результат поразрядного циклического сложения
	чед	результат поразрядного циклического сложения количества единиц в значении

		левого операнда со значением правого операнда
	нед	результат поразрядного циклического сложения номера старшей единицы левого значения с правым значением
	сдв	значение левого операнда сдвигается на количество разрядов, определяемое значением правого операнда;если правое значение положительно,то происходит сдвиг вправо,если отрицательно- то влево.В том случае когда абсолютная величина правого значения превышает 48, результат не определен
	сдвп	операция полностью эквивалентна операции сдв
	сдвл	операция обратна операции сдв в том смысле,что сдвиг происходит влево при положительном значении правого операнда и вправо, если значение отрицательно

Использование дополнительных операций в ряде случаев может повысить эффективность программы. Так, например, процедура СТЕПЕНЬ из шестой главы, написанная с использованием операции сдвл , будет выглядеть значительно проще :

```
СТЕПЕНЬ (X)=I сдвл X;
```

Множество дополнительных операций легко отображается в систему команд ЭВМ БЭСМ-6. Для непосредственного использования системы команд язык предоставляет специальное средство.

8.4. Команды

```
операция ::= команда
команда ::=÷!?восьмеричная-цифра ! (!:*)? :
```

Команды, как видно из синтаксиса, занимают позицию бинарной операции. Команды БЭСМ-6 можно рассматривать как бинарные операции, левым операндом которых является содержимое сумматора, а правый определяется адресом команды. Эта аналогия с бинарной операцией поддерживается в языке следующим образом. Исполнение произвольной команды состоит из нескольких этапов, в результате которых перед непосредственным исполнением команды значение левого операнда оказывается вычисленным и размещенным на сумматоре, а правый операнд должен быть представлен адресом. После выполнения (в случае необходимости) описанных действий выполняется команда, составленная из восьмеричного кода операции и сформированного адреса. Адрес размещается, начиная с первого разряда команды. Код операции размещается с I3-го разряда команды путем логического сложения. Из этого следует, что к коду длинноадресной команды должен быть справа приписан ноль. Команда может сопровождаться стрелкой. Стрелка, расположенная слева (справа), служит указанием транслятору для расположения команды в левой (правой) половине ячейки. Наличие звездочки при команде в том случае, когда она попадает в левую половину ячейки, вызывает

заполнение правой половины командой, не влияющей на результат исполнения программы.

8.5. Сегменты

Для разбиения объектного кода программы на части в языке существуют средства сегментации. Любую процедуру, модуль или метку можно отнести к некоторому сегменту. При этом весь объектный код разбивается на несколько частей, одна из которых называется резидентной и составлена из программных фрагментов, не попавших ни в один описанный сегмент. Для взаимодействия сегментов может существовать несколько стандартных стратегий, предлагаемых системой программирования. Кроме того, существует возможность программирования любой индивидуальной стратегии.

При описании сегментов с именем сегмента может быть связана либо адресная пара, указывающая начальный адрес сегмента, либо имя ранее определенного сегмента, либо указание о принадлежности сегмента к резиденту.

описание-сегментов ::= сегменты (описание-сегмента/;)
описание-сегмента ::= имя%сегмента% = (адресная-пара :
    имя%ранее определенного сегмента% : резидент)

Описание сегмента дает возможность отнести к этому сегменту любую процедуру или квазипроцедуру путем включения в соответствующую анкету имени сегмента.

раздел-анкеты% (виртуальной или квази-) процедуры% ::=
    имя%сегмента%

Объектный код модуля разбивается на две части. Первая часть порождается процедурами и квазипроцедурами, описанными внутри модуля. Вторая часть является образом вход-предложения и инициализации переменных. Анкета модуля позволяет указать различные сегменты для этих двух частей.

описание-модуля ::=
    имя%модуля% (* анкета%модуля%)?
    (((формальный-параметр /,)))?=блок
раздел-анкеты%модуля% ::=
    имя%сегмента% : иниц =имя%сегмента%

Раздел анкеты, указанный просто именем сегмента, предписывает отнести все квазипроцедуры и процедуры модуля к указанному сегменту. Исключение составляют те процедуры , в анкетах которых имеется указание о сегменте.

К сегменту, помеченному иниц , относится вышеупомянутая вторая часть модуля.

В любом месте программы всегда известен "текущий" сегмент, которому будет принадлежать соответствующий фрагмент объектного кода. Именно этот сегмент можно опускать в соответствующих разделах анкеты.

Динамические и модульные блоки, отнесенные к одному и тому же сегменту, располагаются в одном месте (которое может меняться в процессе исполнения программы) вместе со своими общими данными (например, полем констант).

Для того, чтобы несоответствие сегментной и логической структур не нарушало целостности исполнения программы, принимаются некоторые меры, которые касаются всех четырех полей объектной программы: команд, переменных, строк и констант.

Для обеспечения вызовов подпрограмм из произвольных сегментов предполагаются, как уже упоминалось, некоторые стандартные стратегии.

Все переменные вне динамических блоков, не адресуемые явно, относятся к резидентному сегменту. Все строки составляют отдельный сегмент.

Сложнее всего дело обстоит с константами, при размещении которых возможны различные варианты. Для того, чтобы дать возможность указывать вариант размещения константы, вводится анкета при описании константы.

описание-константы ::= имя%константы%
(* анкета%константы%)?
(: запись)?= (константа /,)
раздел-анкеты%константы% ::= имя%сегмента% : уник

Имя сегмента в анкете указывает место первой прописки константы. Прописка константы в некотором сегменте означает, что она, во-первых, будет размещена в поле констант данного сегмента и, во-вторых, при всех использованиях в сегменте будет представлена адресом размещения в этом сегменте. Отсутствие имени сегмента в анкете означает, что константа будет прописана в текущем сегменте.

При использовании константы в сегменте, в котором она не прописана, возможны два случая. Если в анкете константы не была указана ее уникальность (уник), то константа будет прописываться во всех использующих ее сегментах. Наличие раздела уник означает, что константа имеет единственную прописку, полученную при описании.

8.6. Атрибуты

Для вычисления адресов объектов в языке существует унарная операция (адрес). Она позволяет вычислять адреса таких объектов, как константы и переменные. Для выяснения адресов программного кода можно обращаться к именам процедур и модулей. Для того, чтобы обратиться к произвольной части программного кода, нужно пометить соответствующую формулу. Все метки должны быть описаны, а сегмент из анкеты должен совпадать с тем, в котором метка будет определена. Доступность имен меток подчиняется общим правилам. Таким образом, синтаксис пополняется следующими правилами:

описание-меток ::= метки (описание-метки/;)
описание-метки ::=имя%метки% (* анкета%метки%)?
раздел-анкеты%метки% ::= имя%сегмента%

предложение ::= ((имя%метки% :)? формула /;)
первичное ::= адрес (имя%константы или переменной%
(. селектор-записи)?: имя%метки, процедуры,
квазипроцедуры или модуля% :
параметр (константа)?)

Значением такого первичного является фактический адрес объекта в момент исполнения операции.

Адрес объекта выражается с помощью определенной адресной пары. Эта связь постоянна на протяжении всей программы. С помощью так называемых атрибутов язык позволяет обращаться к индексному регистру (ир) и смещению (смещ), определяемым адресной парой. Как видно из приводимого ниже синтаксиса, с объектами могут быть связаны и другие атрибуты.

атрибут ::= (ир : смещ)
(имя%константы, переменной% (. селектор записи)?:
имя%метки, сегмента, (квази)процедуры%)?:
(ирдан : смещдан)
имя%процедуры или квазипроцедуры% ?:
размер (имя%записи, константы, переменной%
(.селектор-записи)?:
имя%процедуры или квазипроцедуры%)?:
(размлок : размпар : числпар)
имя%процедуры или квазипроцедуры%? :
номсег имя %сегмента или (квази) процедуры%? :
(прагр : легр) имя%записи% (.селектор записи)?

Отсутствие имени при атрибуте возможно в предложениях, определяющих класс. Это означает, что при подстановках класса будет подставляться атрибут соответствующей квазипроцедуры.

Размер указывает число ячеек, отводимых под константу, переменную, вектор, программное поле процедуры или квазипроцедуры. С динамическими блоками связаны такие атрибуты, как размер области, отводимой под локальные объекты (размлок), размер области, отводимой

только под параметры (размпар), а также число параметров (чиспар).

В результате трансляции составляется таблица сегментов, вид которой определяется реализацией. По имени сегмента или принадлежащей ему процедуры (квазипроцедуры) можно узнать его номер в таблице (номсег). Каждая конкретная реализация языка может предусмотреть расширенный набор атрибутов.

Имеется возможность узнавать индексные регистры и смещения для констант и переменных с помощью атрибутов ир и смещ . Для программных единиц эти атрибуты указывают координаты объектного кода. Для информации о локальных переменных имеются атрибуты ирдан и смещдан .

Атрибуты прагр и легр дают номера разрядов правой и левой границ для соответствующего поля записи.

8.7. Постоянные выражения

Значения выражений, состоящие из атрибутов и других констант, являются постоянными во все время исполнения программы и могут быть в принципе вычислены на стадии трансляции. Для того, чтобы облегчить реализацию такой возможности, в языке существует специальный сорт констант.

```
константа ::= постоянное-выражение : атрибут
постоянное-выражение ::=
    ( операнд-постоянного-выражения / операция )
операнд-постоянного-выражения ::=
    изображение-значения :
    имя%константы% (. селектор-записи )?:
    атрибут : составное
составное ::=
    запись [(операнд-постоянного-выражения /,)]
```

8.8. Примеры

После введения средств прямого использования вычис-

лителя можно на примерах проиллюстрировать задание произвольных структур данных с указанием алгоритмов доступа (см. 7.2), а также описание классов процедур с программным заданием их реализации (см. 7.3).

8.8.1. Примеры доступов

Ниже показывается задание одной и той же структуры данных как с помощью стандартного средства – описания вектора, так и посредством описания соответствующего доступа.

Пусть имеются следующие описания векторов:

```
1) переменные ВI[100];
2) переменные /в 30000/ = В2[100];
3) переменные /в 20000/ = В3[100]:ЗАП1;
```

Доступы, эквивалентные в смысле результата, могут выглядеть следующим образом:

```
1) доступ ДI[I]="(I-1)↑BI"→
2) переменная /в 30000/ = ПЕР:/100/;
   доступ Д2[I]="(I-1)↑ПЕР"→
3) доступ Д3[I]=
        "(I-1x2+в 20000)↑ЗАП1"→ ;
```

Теперь приведем пример описания доступа, определяющего структуру данных, которую неудобно задавать посредством описания вектора. Примером такой структуры может служить плотный байтовый массив. Помещенное ниже описание доступа МАССИВ задает алгоритм чтения байта из такого массива.

```
переменные    ПОЛЕ:/1000/;
              N;
доступ МАССИВ[I]=
              "((I-1 дел 6)↑ПОЛЕ→N;
```

```
выбор I-(I-I дел 6 х6)
из N * [4I:48]
 , N * [33:40]
  , N * [25:32]
  , N * [I7:24]
  , N * [9:I6]
  , N * [I:8]
все )"
```

Следует отметить, что примеры в данном описании приводятся для иллюстрации средств языка, без обсуждения условий применения на практике.

Так, например, байтовый массив допускает различные варианты программирования, один из которых приведен ниже.

```
МАССИВ [I]="(I-I дел 6->N) ↑ ПОЛЕ сдв (N+Iх6-Iх8) ∧ ш FF"
```

8.8.2. Примеры классов квазипроцедур

Каждый из приводимых ниже примеров включает в себя описание класса, задающее некоторый способ передачи параметров процедуры, и описание квазипроцедуры, функционирование которой обеспечивается этим описанием класса.

8.8.2.I. Передача параметров по значению

```
начало
      класс ПЕРЕДАЧА ПАРАМЕТРОВ ПО ЗНАЧЕНИЮ=

        "(÷250:/ в I7,2/;
        << параметр -> / в I7,0/;>>
        ÷220:/ ир , 0/÷3I0↑:/ в I6, смещ /)"
                    :
        "÷250:/ в I7, в 77776- чиспар /
       ÷42:/0, в I6/÷43:/0, в I5/-> / в I7,0/
      ÷44:/ в I7,  в I5/÷250:/ в I7, размлок/
                  тело
       ÷44:/ в I5, в I7/÷230:/ в I7,0/
       ÷240:/ в I5,0/÷230:/ в I7,0/÷300:/0/";
```



```
квазипроцедура ФАКТОРИАЛ * [ ПЕРЕДАЧА ПАРАМЕТРОВ ПО
                             ЗНАЧЕНИЮ, чиспар =I]=
        начало
              доступ X="/ в I5,0/"->;
        вход
           если X>I
           то ФАКТОРИАЛ (X-I)хX
           иначе   I
          все
      конец
  ..........................................
вход
    ...; ФАКТОРИАЛ (3);...
конец
```

8.8.2.2. Передача параметров по ссылке

```
начало
      класс    ПЕРЕДАЧА ПАРАМЕТРОВ ПО ССЫЛКЕ =
          "÷250:/ в I7, 2/;
     <<÷220:/ ир параметр , смещ параметр /÷240:
          / в I6,0/÷42:/ в I6/->/ в I7,0/;>>
         ÷220:/ ир , 0/÷3I0↑:/ в I6, смещ /"
              :
          "÷250:/ в I7,  в 77776- чиспар /
       ÷42:/0, в I6/÷43:/0, в I5/-> / в I7,0/
        ÷44:/ в I7,  в I5/÷250:/ в I7, чиспар /;
         тело  ;
          ÷44:/ в I5,  в I7/÷230:/ в I7,0/
          ÷240:/ в I5,0/÷230:/ в I7,0/÷300:/0/";
квазипроцедура СКАЛ УМН ВЕКТ * [ПЕРЕДАЧА
            ПАРАМЕТРОВ ПО ССЫЛКЕ, чиспар =3]=
     начало доступы X[I]="(/ в I5,0/+I-I)↑"->;
                    У[I]="(/ в I5,I/+I-I)↑"->;
                    N="/ в I5,2/↑"->;
          переменные J; S;
     вход 0->J->S;
```

```
                пока I↦J ⩽ N
              цикл X[J]xУ[J]↦S
                   иначе S
             все
            конец ;
 переменные BI[I:IO];
      B2[I:IO];
         M;
 ..........................................
    вход
   ...; если размер BI→M= размер B2
        то  СКАЛ УМН ВЕКТ (BI, B2, M )
        все   ; ...
    конец
```

8.8.2.3. Передача параметров по имени

```
начало
  класс  ПЕРЕДАЧА ПАРАМЕТРОВ ПО ИМЕНИ =
     "÷250:/ в I7,2/;
      << начало метка M;
вход ÷220:/ ир M,0/÷3IO↑:/ в I6, смещ M/
     ÷42:/0, в I4/
     ÷43:/0, в I6/÷43:/0, в I5/→/ в I7,0/
     ÷230:/ в I5, в 77777/÷240:/ в I5,0/;
     ÷220:/ в I7,0/÷240:/ в I4, в 77775/;

       параметр ;

     ÷230:/ в I4,2/÷240:/ в I5,0/
     ÷44:/ в I4, в I7/
     ÷230:/ в I4,0/÷240:/ в I4,0/
     ÷230:/ в I7,I/÷300:/0/;
     M: 42:/0, в I6/→/ в I7,0/

конец >>
```



```
÷220:/ ир ,0/÷3IO↑:/ в I6, смещ /"
    :
"÷250:/ в I7, в 77776- чиспар /
     ÷42:/0, в I6/÷43:/0, в I5/→/ в I7,0/
    ÷44:/ в I7, в I5/÷250:/ в I7,  чиспар /;
    тело ;
    ÷44:/ в I5, в I7/÷230:/ в I7, 0/
    ÷240:/ в I5,0/÷230:/ в I7,0/÷300:/0/";
квазипроцедура МАКСИМУМ
                * [ПЕРЕДАЧА ПАРАМЕТРОВ ПО ИМЕНИ,
                   чиспар =2]=
                    начало доступы
                  X="÷230:/ в I5,0/÷3IO↑:/ в I6,0/";
                  У="÷230:/ в I5,I/÷3IO↑:/ в I6,0/";
                    вход
                          если X⩾У
                          то X
                          иначе У
                          все
                    конец ;
                    переменные A;B;C;
                  ..........................................
вход
...;МАКСИМУМ (A+Bx2, если A≠B то 2 иначе C все );...
конец ;
```

СИНТАКСИС ЯЗЫКА ЯРМО

```
адрес (3, 8.I, 8.2) ::=переменная :
    адресная-пара : косвенность
адресная-пара (8.I) ::=
    ⌊ (константа%индекс%,)? константа%смещение% ⌊
альтернативное (4.I) ::=
    условное : выбирающее : перебирающее
анкета (5.3) ::=[(раздел-анкеты /,)]
атрибут (8.6) ::=
    ( ир : смещ )(имя%константы или переменной%
    (.селектор-записи)?: имя %метки, сегмента,
    процедуры или квазипроцедуры %)?:
    ( ирдан : смещдан ) имя%процедуры или
    квазипроцедуры%?:
    размер (имя%записи, константы, переменной%
    (.селектор-записи)?:
    имя%процедуры или квазипроцедуры%?:
    ( размлок : размпар : чиспар)
    имя%процедуры или квазипроцедуры% :
    номсег имя%сегмента, процедуры или
    квазипроцедуры%?:
    ( прагр : легр )имя%записи%
    (.селектор-записи)?
блок (5) ::= начало описания вход предложение конец
буква (2.I.3) ::=
    А:Б:В:Г:Д:Е:Ж:З:И:Й:К:Л:М:Н:О:П:Р:
    С:Т:У:Ф:Х:Ц:Ч:Ш:Щ:Ы:Ь:Э:Ю:Я:D:F:G:
    I:J:L:N:Q:R:S:U:V:W:Z
восьмеричная-цифра (2.I.I) ::= 0:I:2:3:4:5:6:7
восьмеричный (2.I.2) ::=
    ( в : вл ) восьмеричная-цифра !
выбирающее (4.I) ::=
    выбор предложение
```



```
    из (предложение /,)
    ( иначе предложение)?
    все
выделение-слога (3.2.I) ::=*запись
вызов-квазипроцедуры (7.3) ::=
    имя%квазипроцедуры% ( ⌊ (текст%фактического
    параметра квазипроцедуры% /,) ⌋ )?
вызов-модуля (5.3) ::=
    имя%модуля% ( ⌊ ( предложение /,) ⌋ )?
вызов-процедуры (5.2) ::= имя%процедуры%
    ( ⌊ (предложение /,) ⌋ )?
вызов-текста (7.I) ::= имя%текста% ( ⌊ (текст /,) ⌋ )?
граница (2.I.3, 2.2.I) ::= постоянное-выражение
группа-параллельных-полей (2.I.3, 2.2.I) ::=[(поле /,)]
группа-последовательных-полей (2.2.I) ::= ⌊ (поле /,) ⌋
двоичный (2.I.2) ::= ( д : дл )(0 : I)!
завершитель (4.3, 5.2) ::=
    завершить имя%уровня%? ( при условие все )?:
    возврат ( при условие все )?
запись (2.I.3, 2.2.I) ::=
    имя%записи% :
    изображение-записи
засылка (3.3) ::=
    простая-засылка : совмещенная-засылка
изменение-слога (3.2.2) ::= ⇒запись * первичное
изменение-слога-с-засылкой (3.2.2) ::=
    -> запись * адрес
изображение-записи (2.I.3, 2.2.I) ::=
    ⌊ размер-записи ⌊ :
    ⌊ множитель ⌊ x компонента-массива :
    группа-параллельных-полей :
    группа-последовательных-полей
изображение-значения (2.I, 3.I) ::=
    число : набор : составное :
    строка : истина : ложь
имя (2.I.3) ::= буква : ( буква : цифра )!?
команда (8.4) ::=÷↑?восьмеричная-цифра ! (↑:*)? :
компонента-массива (2.2.I) ::= запись
```

```
константа (2.2.I, 2.2.2, 8.7) ::=
    простая-константа : постоянное-выражение
косвенность (8.2) ::= первичное ↑ (адресная-пара :
    имя%константы или переменной% :
    имя%записи% (. селектор-записи)?)?
множитель (2.2.I) ::= постоянное-выражение
набор (2.I.2) ::= шестнадцатеричный :
    восьмеричный : двоичный : текстовый
операнд-постоянного-выражения (8.7) ::=
    изображение-значения :
    имя%константы% (. селектор-записи )?:
    атрибут : составное
операция (3.I, 8.3, 8.4) ::= + : - : x : дел :
    сл : вч : умн : делр : < : ⩽ : > : ⩾ :
    м : мр : б : бр : = : ≠ : ∨ : ∧ : вчаб : вчабр :
    вчоб : вчобр : нтж : чед : нед : слш : сдв :
    сдвп : сдвл
описание (5) ::= описание-записей :
    описание-констант : описание-переменных :
    описание-процедур : описание-модулей :
    описание-текстов : описание-доступов :
    описание-классов : описание-квазипроцедур :
    описание-сегментов : описание-меток
описание-виртуального-доступа (7.2) ::=
    имя%доступа% (* анкета%текста% )?
описание-виртуального-класса (7.3) ::=
    имя%класса%
    (* анкета%текста% )?
описание-виртуального-текста (7.I) ::=
    имя%текста% (* анкета%текста% )?
описание-виртуальной-квазипроцедуры (7.3, 8.I) ::=
    (адресная-пара≡)?
    имя%квазипроцедуры% (* анкета%квазипроцедуры% )?
описание-виртуальной-процедуры (5.3, 8.I) ::=
    ( адресная-пара≡)? имя%процедуры%
    (* анкета%виртуальной процедуры% )?
описание-доступа (7.2) ::= имя%доступа%
    ([( имя%формального параметра доступа% /,)])? =
```



```
    ("текст"→ "текст": "текст"→ :"текст":→ "текст")
описание-доступов (7.2) ::=
    общие ? доступы ( описание-доступа /;):
    виртуальные доступы
    (описание-виртуального-доступа /;)
описание-записей (2.I.3) ::=
    записи ( описание-записи /;)
описание-записи (2.I.3) ::= имя%записи% = запись
описание-квазипроцедур (7.3) ::=
    общие? квазипроцедуры (описание-квазипроцедуры/;):
    виртуальные квазипроцедуры
    ( описание-виртуальной-квазипроцедуры /;)
описание-квазипроцедуры (7.3, 8.I) ::=
    (адресная-пара≡)? имя%квазипроцедуры%
    ( * анкета%квазипроцедуры% )?= формула
описание-класса (7.3) ::= имя%класса%
    ( ( ( имя%формального параметра класса% /,) ) )?
    ="текст%программирования вызова%"
    : "текст%программирования входа и выхода%"
описание-классов (7.3) ::=
    общие? классы ( описание-класса /;):
    виртуальные классы (описание-виртуального-класса/;)
описание-констант (2.2.2) ::=
    константы (описание-константы /;)
описание-константы (2.2.2, 8.5) ::=имя%константы%
    (* анкета%константы% )?( : запись )?=
    (простая константа /,)
описание-метки (8.6) ::=имя%метки% ( * анкета%метки% )?
описание-меток (8.6) ::= метки ( описание-метки /;)
описание-модулей (5.3) ::=
    модули ( описание-модуля /;)
описание-модуля (5.3, 8.5) ::=
    имя%модуля% (* анкета%модуля% )?
    ( ( (формальный-параметр/,) ) )? = блок
описание-переменной (2.2.3, 2.3.I, 3, 8.I) ::=
    (адресная-пара≡)?
    имя%переменной%
```

```
    ([константа])? ( : запись )? (= (формула /,))?
описание-переменных (2.2.3) ::=
    переменные ( описание-переменной /;)
описание-процедур (5.2, 5.3) ::=
    общие ? процедуры ( описание-процедуры /;):
    виртуальные процедуры
    ( описание-виртуальной-процедуры /;)
описание-процедуры (5.2, 8.1) ::=
    (адресная-пара≡)?имя%процедуры%
    (* анкета%процедуры% )?
    ( ( (формальный-параметр /,) ) )?= формула
описание-сегмента (8.5) ::=
    имя%сегмента% = (адресная-пара :
    имя%ранее определенного сегмента% : резидент )
описание-сегментов (8.5) ::=
    сегменты ( описание-сегмента /;)
описание-текста (7.1) ::= имя%текста%
    ( ( (имя%формального параметра текста% /,) ) )?=
    "текст"
описание-текстов (7.1) ::=
    общие ? тексты ( описание-текста /;):
    виртуальные тексты (описание-виртуального-текста/;)
описания (5) ::= ( описание /;)
отрицательное (2.1.1) ::=-положительное
первичное (3, 4, 4.3, 5.1, 5.2, 5.3, 7.1, 7.2, 7.3,
8.6) ::=
    простая-константа : адрес :
    ( : имя%уровня% : )?
    (альтернативное : циклическое : ( предложение ) :
    блок) : вызов-процедуры : вызов-модуля:
    вызов-текста : вызов-квазипроцедуры :
    адрес (имя%константы или переменной%
    (. селектор-записи )? :
    имя%метки, процедуры, квазипроцедуры или модуля% :
    параметр ( константа )?): тело .
    повторитель : имя%формального параметра%
перебирающее (4.1) ::=
    перебор предложение
```



```
    из ((предложение /,) то предложение /,)
    (иначе предложение )?
    все
 переменная (2.2.3, 2.3.1, 7.2) ::= имя%переменной%
    ([предложение])? (.селектор-записи )?:
    имя%доступа% ( (текст /,) )?
повторитель (7.3) ::=  текст
    ([простая-константа : простая-константа])?
поле (2.1.3, 2.2.1) ::=
    имя%поля% ( = (запись : слог ))?:
    изображение-записи : слог
положительное (2.1.1) ::= цифра !
постоянное-выражение (8.7) ::=
    (операнд-постоянного-выражения / операция )
предложение (2.3.1, 3, 4.3, 8.6) ::= (( имя%метки% : )?
    формула завершитель ?/;)
программа (5.1) ::= (: имя%уровня% :)?блок
простая-засылка (3.3) ::= → адрес
простая-константа ::= операнд-постоянного-выражения :
    |постоянное-выражение|
раздел-анкеты%виртуальной процедуры% (5.3, 8.5) ::=
    чиспар = простая-константа : имя%сегмента%
раздел-анкеты%квазипроцедуры% (7.3, 8.5) ::=
    имя%сегмента% :
    ( чиспар : размпар ) = простая-константа :
    имя%класса% ( ( ( текст%фактического параметра
    класса% /,) ) )?
раздел-анкеты%константы% (8.5) ::=
    имя%секмента% : уник
раздел-анкеты%метки% (8.6) ::= имя%сегмента%
раздел-анкеты%модуля% (8.5) ::=
    имя%сегмента% : иниц = имя%сегмента%
раздел-анкеты%процедуры% (8.5) ::= имя%сегмента%
раздел-анкеты%текста% (7.1) ::= чиспар =
    простая-константа
размер-записи (2.2.1) ::= постоянное-выражение
рациональное (2.1.1) ::=
    р-?((положительное ?. положительное :
```

```
    положительное)(10 целое)?:10целое)
селектор-записи (2.2.I) ::= (изображение-значения :
    имя%поля% )(. селектор-записи )?
слог (2.I.3, 2.2.I) ::= граница : граница
совмещенная-засылка (3.3) ::= операция→ адрес :
    *(→ запись * адрес :
    ⇒ запись * первичное) :
    операция→ запись * адрес
составное (2, I.3, 8.7) ::=
    запись [( операнд-постоянного-выражения /,)]
строка (2.I.4) ::="литера !?"
текст (7.I) ::= предложение
текстовый (2.I.2) ::= ( т : тл)"литера !"
условие (4.I, 8.I) ::= предложение ( ист : неист :
    ( ирноль : ирнеполь ) простая константа )?
условное (4.I) ::= если условие
    то предложение
    ( иначе предложение )?
    все
формальный-параметр (5.2) ::=
    имя%переменной% ( : запись )?
формула (3) ::= первичное ? (операция первичное :
    выделение-слога : изменение-слога :
    изменение-слога-с-засылкой :
    засылка : команда адрес )!?
целое (2.I.I) ::= положительное : отрицательное
циклическое (4.2, 8.I) ::=
    пока условие
    цикл предложение
    ( иначе предложение )?
    все :
    повторить предложение
    ( когда условие
    ( иначе предложение )?)?
    все :
    по простая-константа
    ( раз предложение )?
    цикл предложение
```



```
    ( иначе предложение )?
    все
цифра (2.I.I) ::= восьмеричная-цифра : 8 : 9
число (2.I.I) ::= целое : рациональное
шестнадцатеричный (2.I.2) ::=
    ( ш : шл )(цифра : A : B : C : D : E : F )!
```

Оглавление.

В.И.Гололобов, Б.Г.Чеблаков, Г.Д.Чинин

ОПИСАНИЕ ЯЗЫКА ЯРМО
МАКРОСРЕДСТВА

Препринт
248

Технический редактор В.С.Сергеев
Художник-оформитель И.Г.Бархатова

Подписано в печать 29.10.80. МН 06994
Формат бумаги 60x90 1/16 Объем 2.7 уч.-изд.л., 2.9 п.л.
Тираж 250 экз. Заказ 453

Ротапринт ВЦ СО АН СССР, Новосибирск 90